Asynchronous Transfer Mode (ATM) Conversion Device (ACD) System Description Document

August 1997

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland

ASYNCHRONOUS TRANSFER MODE (ATM) CONVERSION DEVICE (ACD) SYSTEM DESCRIPTION DOCUMENT

August 1997

Prepared by:

C. Harris, Systems Engineer
RMS Technologies, Inc.

Reviewed by:

F. Linehan, Manager, VLSI
RMS Technologies, Inc.

Reviewed by:

J. Werner, Systems Engineer
AlliedSignal Technical Services

Quality Assured by:

J. Klein, Quality Engineer
Unisys Inc.

Edited by:

L. Kane, Sr. Technical Writer
NYMA Inc.

Approved by:

N. Speciale, Head
Microelectronic Systems Branch
Code 521, NASA/GSFC

Goddard Space Flight Center
Greenbelt, Maryland 20771

PREFACE

This document is under the configuration management of the Microelectronic Systems Branch Configuration Control Board (CCB). Changes to this document may be made by Documentation Change Notice (DCN), reflected in text by change bars, or by complete revision.

Requests for copies of this document, along with questions or proposed changes, should be addressed to:

Technology Support Office
Code 520
Goddard Space Flight Center
Greenbelt, Maryland 20771

CHANGE INFORMATION PAGE

List of Effective Pages	
Page Number	**Issue**
Title	Original
Signature Page	Original
iii through x	Original
1-1 thru 1-8	Original
2-1 thru 2-9	Original
3-1 thru 3-13	Original
4-1thru 4-4	Original
5-1 thru 5-18	Original
6-1 thru 6-5	Original
AB-1	Original

Document History		
Issue	**Date**	**DCN No.**
Original	May 1997	

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

TABLE OF CONTENTS (CONT'D)

TABLES

FIGURES

SECTION 1
GENERAL INFORMATION

1.1 DOCUMENT INTRODUCTION

This document describes the Asynchronous Transfer Mode (ATM) Conversion Device (ACD) System. This document describes the ACD System design, structure, and available functions. It includes general information about the system: high-level data flow; theory of operation; description of system components; software overview; hardware overview; system and hardware configuration; user interfaces; and applicable supporting documentation. Supporting documents are listed in Section 7, Reference Documentation.

1.1.1 PURPOSE

The ACD System is an evaluation system for NASA GSFC Code 540. The system is being examined as an option to replace high-cost satellite links that transfer telemetry data between remote sites. This document will accompany the ACD System delivery to Code 540 and serve as a reference.

After evaluation is complete, the ACD System is scheduled for deployment to various sites. Figure 1-1 illustrates the possible implementation sites for the ACD System.

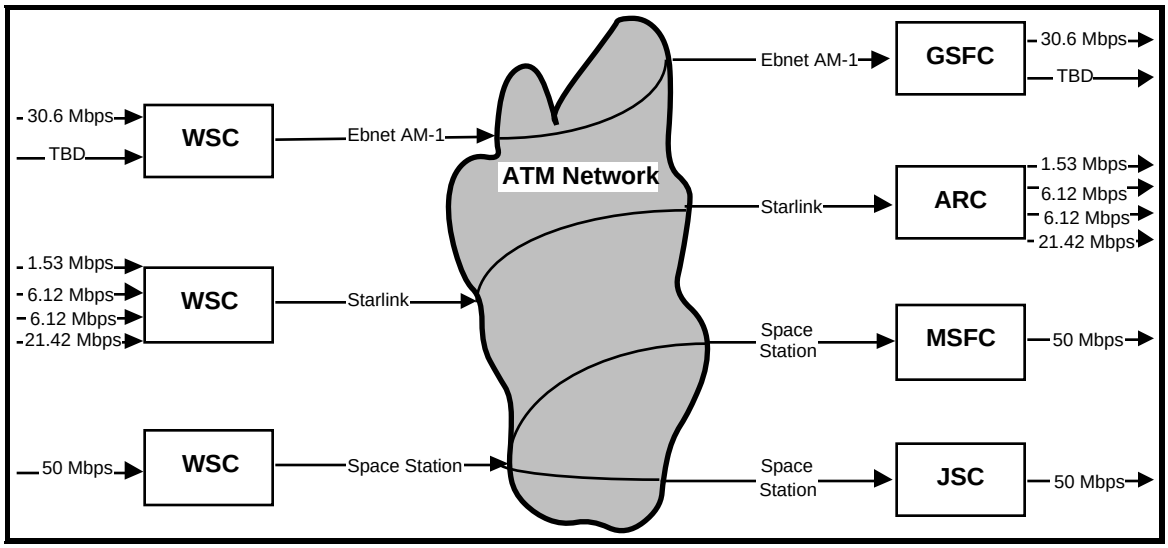


Figure 1-1. Possible ACD System Implementation Plan

1.1.2 SCOPE

This document does not include operator interface details or details of the system-level test plan and procedures.

1.1.3 TERMINOLOGY

It is assumed that the operator has a rudimentary knowledge of ATM. This document provides only an overview of ATM data packaging.

1.2 SYSTEM OVERVIEW

1.2.1 INTRODUCTION

The Asynchronous Transfer Mode (ATM) Conversion Device (ACD) System is based on state-of-the-art ATM technology. It was developed by NASA Code 521 as an option to replace high-cost satellite links that transfer telemetry data between remote sites. The system interfaces between high-rate ECL/RS422 raw data bitstreams and Synchronous Optical Network (SONET) OC-3 fiber. The SONET OC-3 interface uses ATM Adaptation Layer Type Five (AAL5) format.

The system exceeds its 50 Mbps raw data, single stream requirement and provides single stream raw data throughput at rates up to 65 Mbps. With ATM and SONET packaging overhead, this translates into 78 Mbps on the OC-3 fiber.

In addition to high-rate throughput, the system provides multiplexing and demultiplexing of multiple stream throughput based on the ATM cell header Virtual Path and Virtual Channel Identifier (VPI/VCI) values. The system is designed with the flexibility to provide between three and six throughput channels. All of which are multiplexed/demultiplexed to and from the same OC-3 interface. Multiple stream cumulative raw data throughput rates of up to 80 Mbps, or 96 Mbps on the fiber, have successfully run.

The ACD Card Set can be controlled and monitored via the Ethernet SNMP interface. The SNMP operator's interface is a Code 521-designed interface that follows standard SNMP format (see Figure 1-2).

The ACD includes a local operator's interface called OPMAN. The OPMAN interface is accessible through a dumb terminal connected to the system Master Controller Card (MCC) via a phone line. The MCC is the left-most card installed in the chassis.

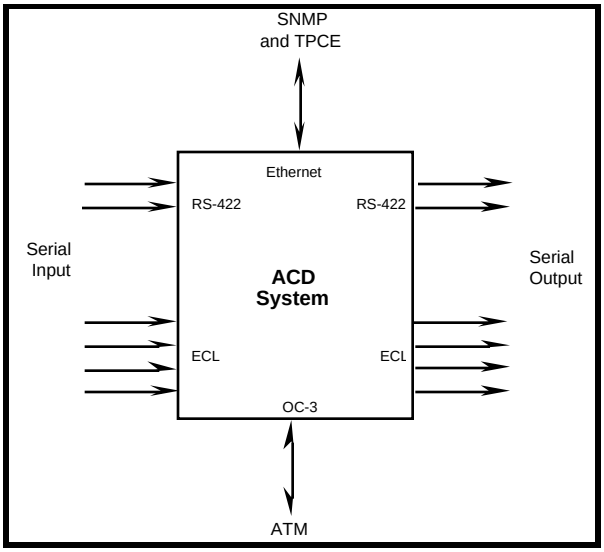


Figure 1-2. ACD Card Set Context Diagram

1.2.2 SYSTEM COMPONENTS

Each ACD card set is an independent system. A split VME backplane is installed in the ACD chassis, thus creating two systems in one chassis. The chassis connects each component (each coponenet is a card) via the VMEbus backplane. In many cases, once an ACD-type system is delivered, either the chassis is slid into a rack, or the chassis shell is removed, and then the system is slid into a rack.

1.2.2.1 Component List

The ACD System component list is broken into three subcategories: ACD card set, ACD chassis, and HC hot-swappable power chassis. One ACD System is comprised of one ACD chassis, which includes ACD card sets, and one or two hot-swappable power supply chassis. The number of power supply chassis coincides with the number of card sets installed in the ACD chassis.

One ACD card set is comprised of the following components:

- **a. (1) MVME 167 32B/32M MCC**
- **b. (1) MVME 167 32B/32M ATM Controller Card**
- **c. (1) MM6290D 32M Memory Card**
- **d. (1) Interphase V/ATM 5215 Adapter Card**
- **e. (1 or 2) SIO Card**
- **f. (1) Seagate SCSI Disk Module**

The major components of one ACD rack-mountable chassis are as follows:

- **a. (2) ten slot backplanes.**
- **b. (1 or 2) ACD card sets.**
- **c. (1) I/O panel that can simultaneously interface to both card sets; includes cables to connect I/O panel to the installed card sets.**
- **d. (1) transition module for each MVME 167 installed; this provides terminal and ethernet connection to each MVME 167.**
- **e. (1) Break-Out-Board (BOB) for each MVME 167 installed in the chassis; this connects each card to the corresponding transition module.**
- **f. (1) acrylic front-door to close in card front panels and encourage correct air flow through the chassis.**

Each hot-swappable power supply chassis includes two HC, 5 channel power supplies. One power supply chassis supports one ACD card set. Therefore, if two card sets are installed in the ACD chassis, the system requires two power supply chassis (see Figure 1-3).

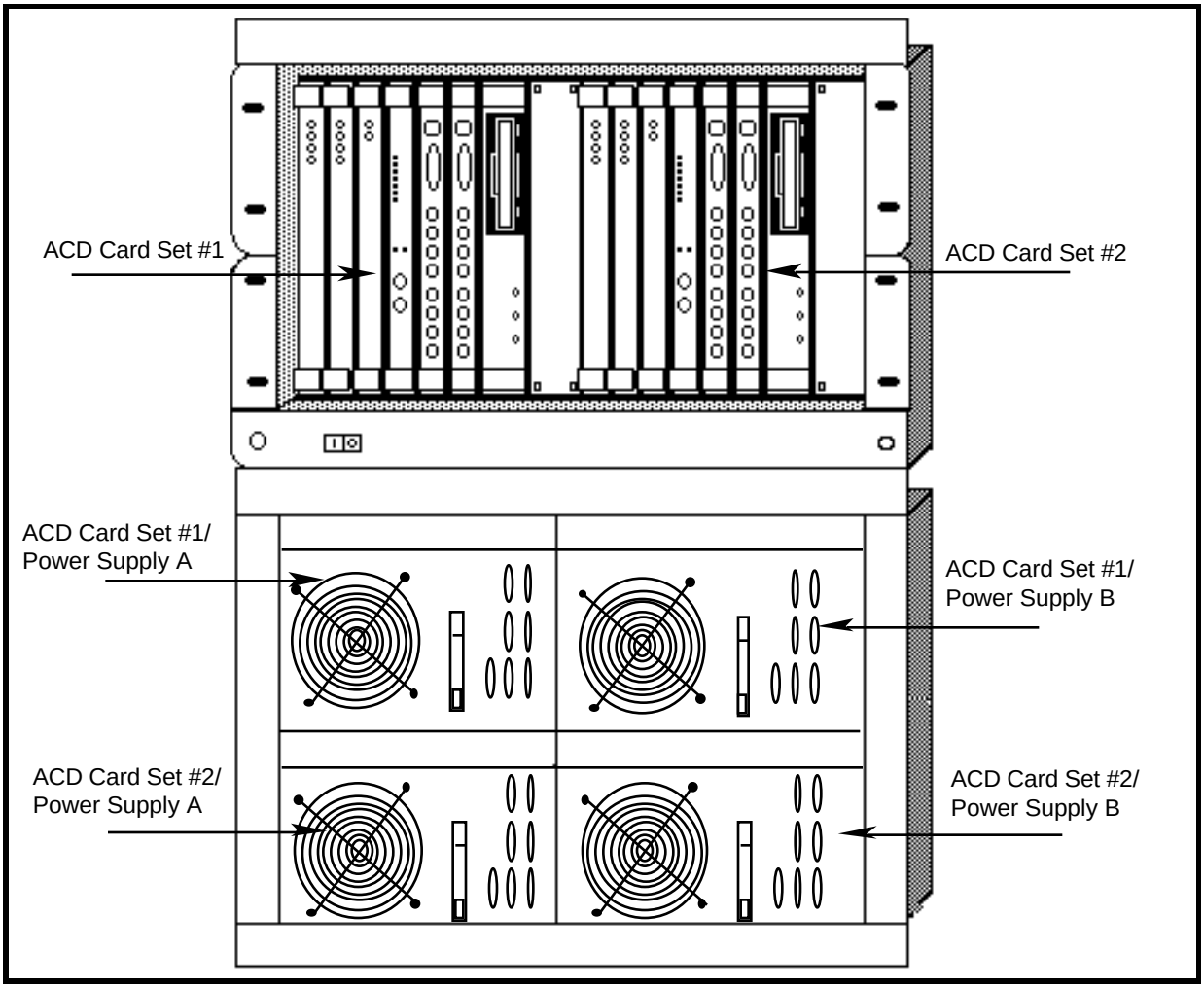


Figure 1-3. ACD Chassis

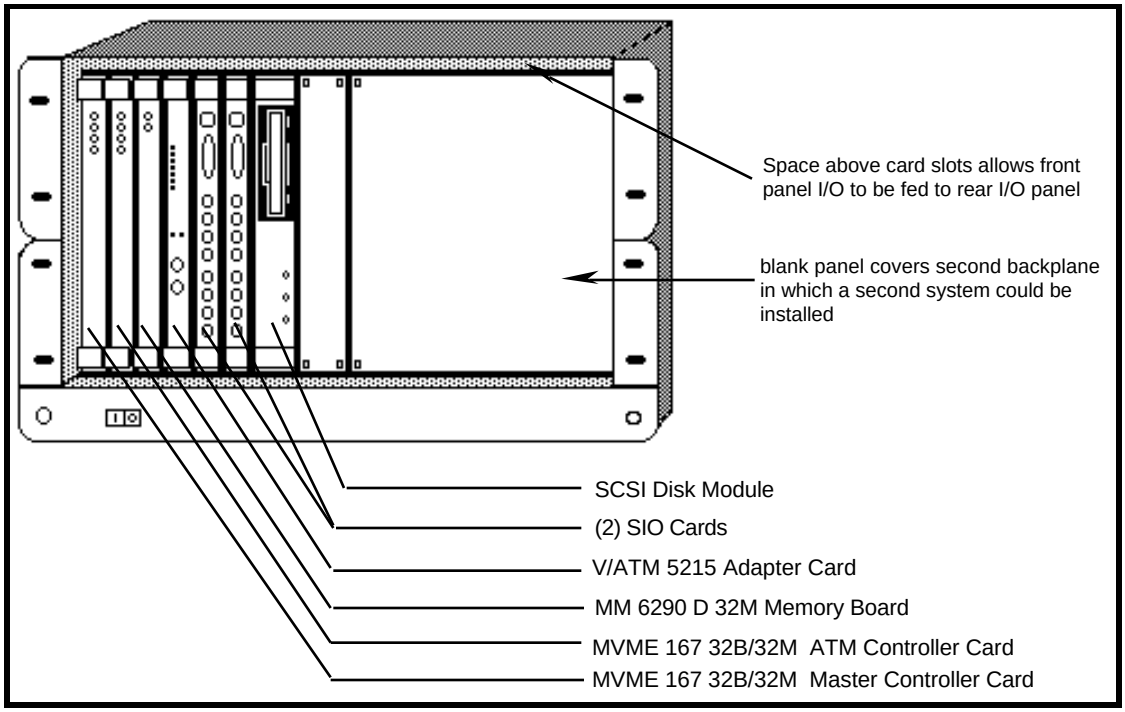


Figure 1-4. One ACD Card Set Installed in One ACD Chassis

1.2.2.2 Chassis Layout and Physical Description

The cards and disk module are connected to the chassis by plugging them into the backplane connectors, and then screwing the front-panels into the chassis. Guides are mounted at the top and bottom of the chassis to line each card up with the backplane connectors and avoid bent pins during card installation.

Ribbon cable, which is not visible unless the back of the chassis is opened, connects the MVME 167 cards (MCC and ATM Controller) to the transition modules that provide the dumb terminal and Ethernet Interfaces.

Data I/O to the system is via a rear I/O panel. Each connector of the rear I/O panel has cables that connect it to the front panel of individual cards.

1.2.3 ACD System Software Overview

The complete ACD System software is the combination of several software packages: OPMAN, subsystem application software, ATM Driver, and SNMP software. SNMP, which is a remote operator interface, can be removed and the system loses no capabilities as long as a dumb terminal is available to run the local operator interface, OPMAN.

The subsystem application software, which is developed in the MEDS environment, is an intrinsic part of system operation and must be present.

Both operator interfaces, SNMP and OPMAN, interface with MEDS. Refer to section 1.2.3.1 for an overview.

1.2.3.1 MEDS, OPMAN, and Subsystem Software

OPMAN and the subsystem application software (MEDS) are running under VxWorks 5.2. MEDS was developed by code 521 to support ACD-type systems (see Figure 1-5). MEDS provides standard libraries, routines, and templates for software development. It also provides a method for communication between the multiple processors in a system and the operator.

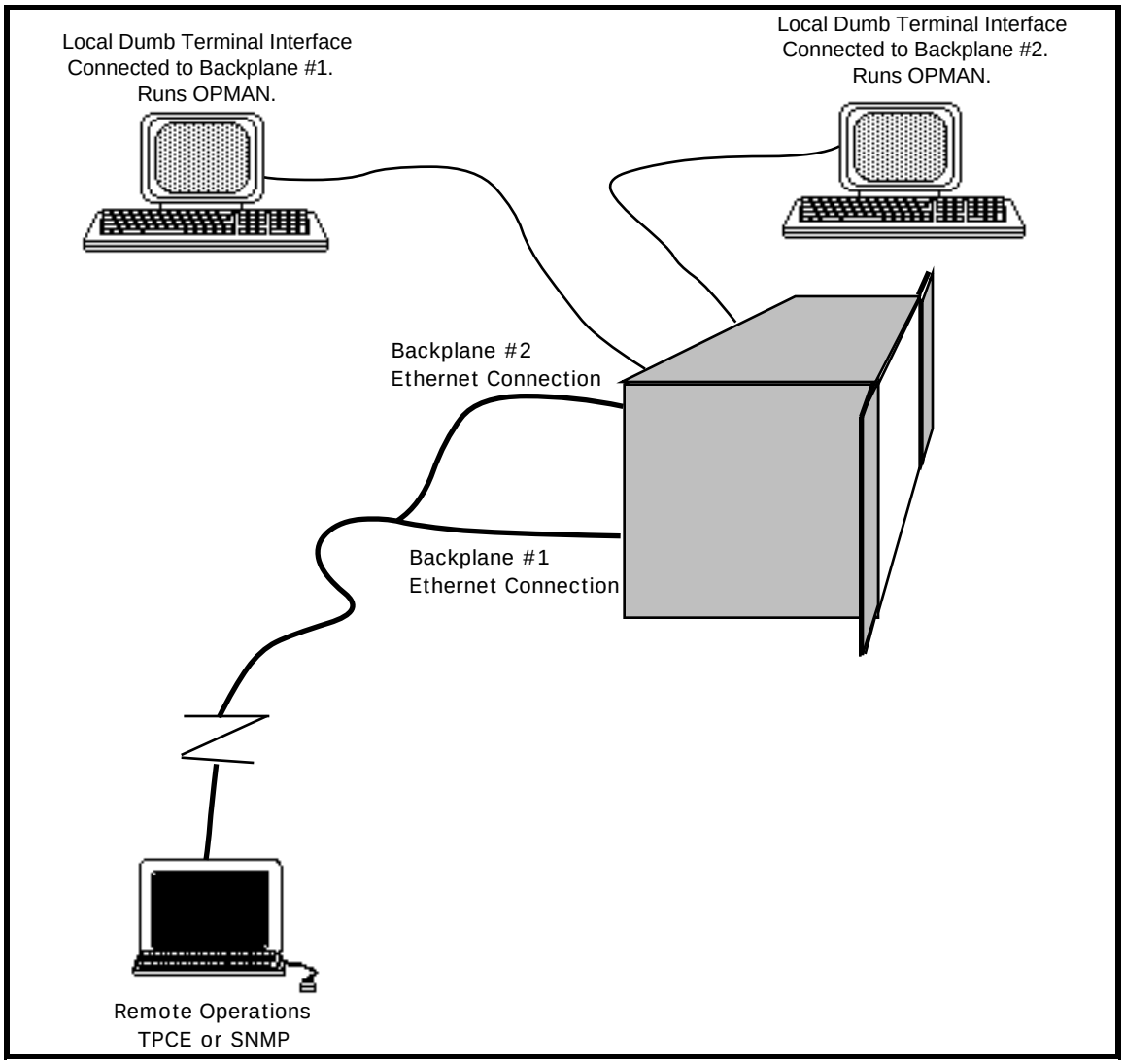


Figure 1-5. ACD System Operator Interfaces

1.2.3.2 SNMP

The ACD System SNMP operator interface is a command line interface, the end user is designing the GUI (see Figure 1-6). SNMP includes all capabilities available on the local operator interface, but it allows multiple ACD Systems to be run from one central point via the network.

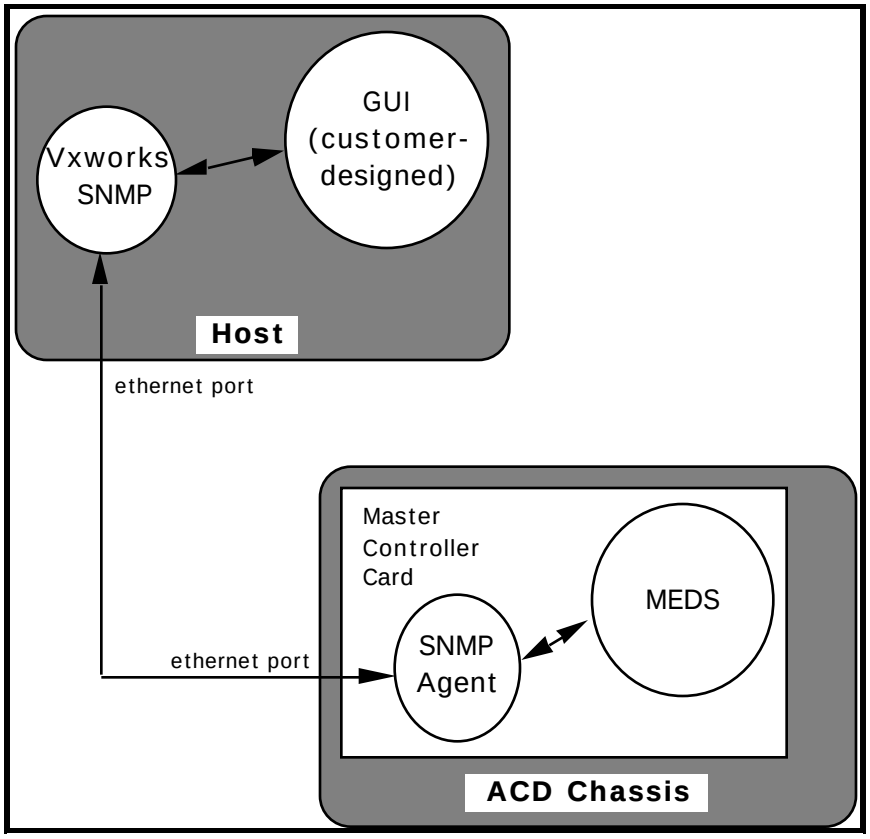


Figure 1-6. SNMP Context Diagram

1.2.4 CARD SET DATA FLOW

Figure illustrates the overall system and all available data flow paths. The number of available data channels directly maps into the number of SIO Cards present in a system. It is expected that many ACD System applications will require only one SIO Card.

It is important to note that Figure 1-7 illustrates all those channels that are available, and each channel can not simultaneously provide input and output. All channels can simultaneously run. However, the number of channels that can run simultaneously is also a variable of system throughput capability. The aggregate system throughput must not exceed 80 Mbps.

For each SIO Card, the ECL data on that card is either input or output, but not both simultaneously. If more than one channel on a card is active, the maximum aggregate data throughput on any one SIO Card is 75 Mbps. Any one ECL channel can run up to 79 Mbps, but simultaneously running SNMP may slow the maximum throughput. Any one RS-422 channel can run up to 10 Mbps.

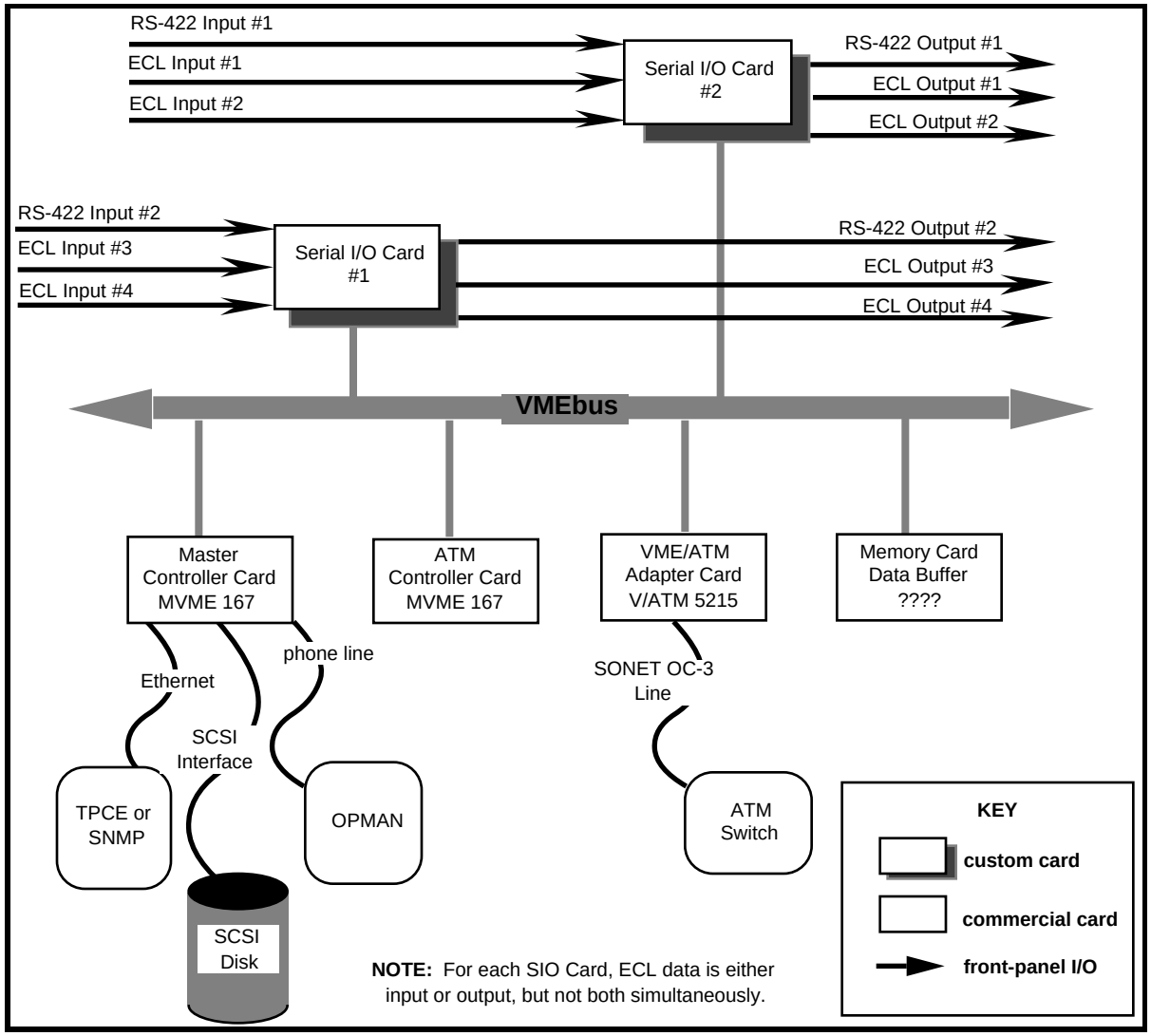


Figure 1-7. ACD Card Set Components and I/O Interfaces

SECTION 2
OPERATING PRINCIPLES

2.1 INTRODUCTION

This section overviews the basic concepts used for ACD Card Set processing. It provides the "theory" of operation, but does not provide the step-by-step procedures of system operation.

2.2 THEORY OF OPERATION

In the ACD Card Set, each component independently performs the functions for which it was designed, and simultaneously reports status and sets indicators to show the state of processing. Each component receives its directions (setup) and reports status to the one central point, which is the Master Controller Card. The local operator interface resides on the MCC, and the MCC is the card with which a remote operator interface communicates. In this way, the translation from operator setup to components, and status reports from components to operator is accomplished.

By setting indicators, which may be a semaphore or an entry in an array, the components can directly communicate with each other without requiring the MCC as an interface. This type of communication is most typically done when components are passing data.

The succeeding subsections detail the ACD primary functions, or modes of operation. A breakdown of functionality by the component that performs it is provided in a later section.

2.2.1 MODES OF OPERATION

The ACD Card Set is either transmitting ATM cells to the OC-3 interface or it is receiving ATM cells from the OC-3 interface. Therefore, for simplicity, the ACD Card Set mode of operation is identified as either Transmit (Input Processing) or Receive (Output Processing).

2.2.1.1 Transmit Mode

The Transmit Mode is often referred to as Input Processing (see Figure 2-1). In this mode, the system function is to receive ECL and/or RS-422 serial input data streams and transfer that data unto an OC-3 line as ATM Cells that contain AAL5 frames. Figure provides an example of Transmit Mode processing.

The Transmit Mode data flow through the system is from one of the SIO Cards to a Memory Card. Once the data is on the Memory Card, the ATM Controller initiates data reads from memory to the V/ATM Card, and then that card formats the data and outputs it to the OC-3 interface. Data transfers from SIO Card to memory, and from memory to the V/ATM Card, are across the VMEbus.

The mode names can become confusing because the different components see the data processing from different perspectives. Specifically, in Transmit Mode, the SIO Card performs input processing (receives front panel input) and the V/ATM Card transmits data. Recognizing the component's perspective is most important when examining SIO Card software documentation that constantly refers to Input Processing, which is the SIO Card's function when in Transmit Mode.

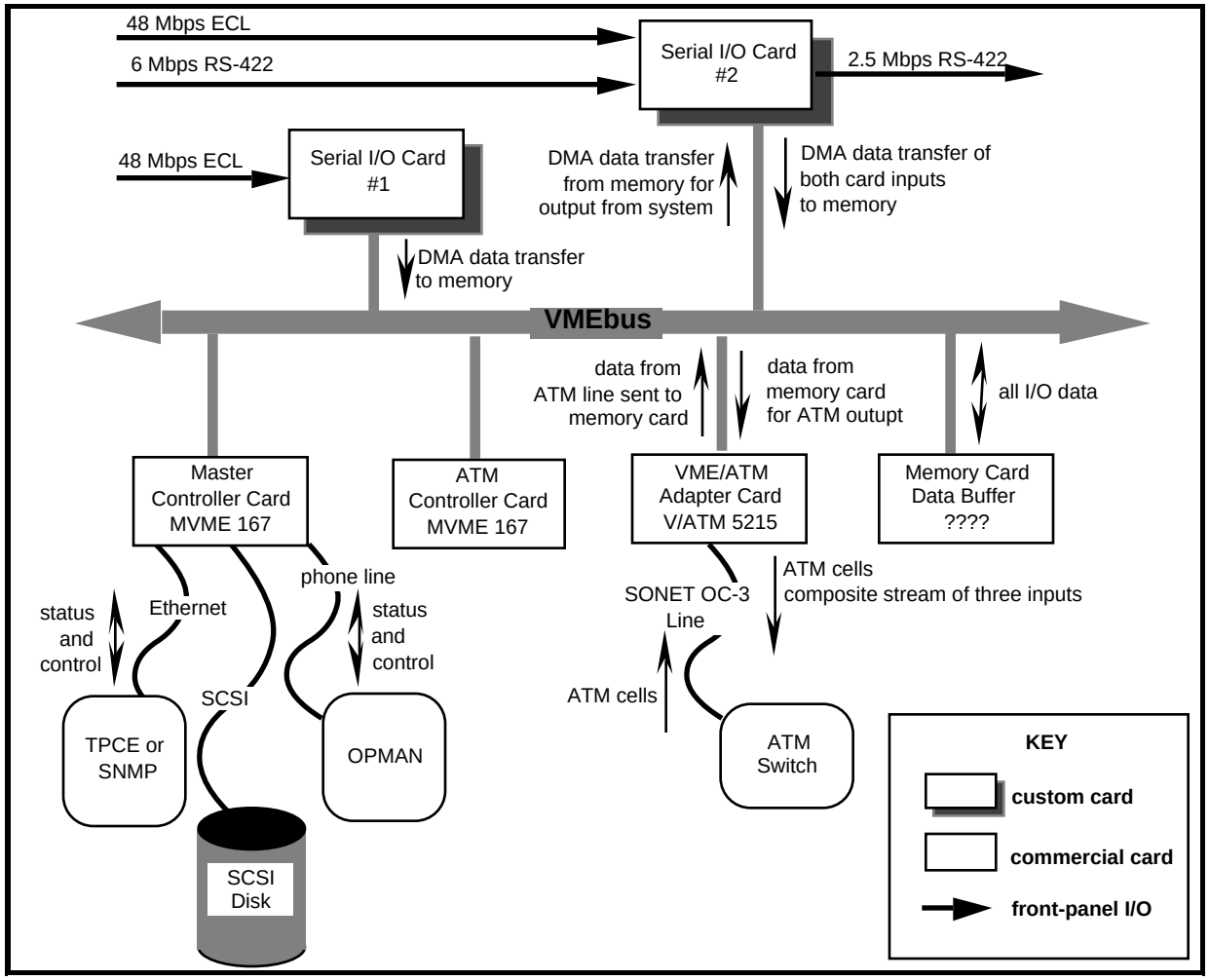


Figure 2-1. Input Processing Data Flow

2.2.1.2 Receive Mode (Output Processing)

Receive Mode accepts ATM Cells from the OC-3 interface, strips ATM and AAL5 packaging from the data, and outputs the data as ECL and/or RS-422 serial streams. Figure 2-2 provides an example of Receive Mode processing.

The Receive Mode data flow through the system is data input to the V/ATM Card, which occurrence is identified by the ATM Controller Card. The ATM Controller Card then initiates a data transfer from the V/ATM Card to the Memory Card and indicates to the SIO Card that data is available for output. Once the data is available, the SIO Card reads the data stored on the Memory Card and outputs it via one of its front panel interfaces (RS-422 or ECL).

Receive Mode is synonymous with Output Processing. Again the dual names result from component perspective. In Receive Mode, the SIO Card outputs data while the V/ATM Card receives data from the OC-3 interface.

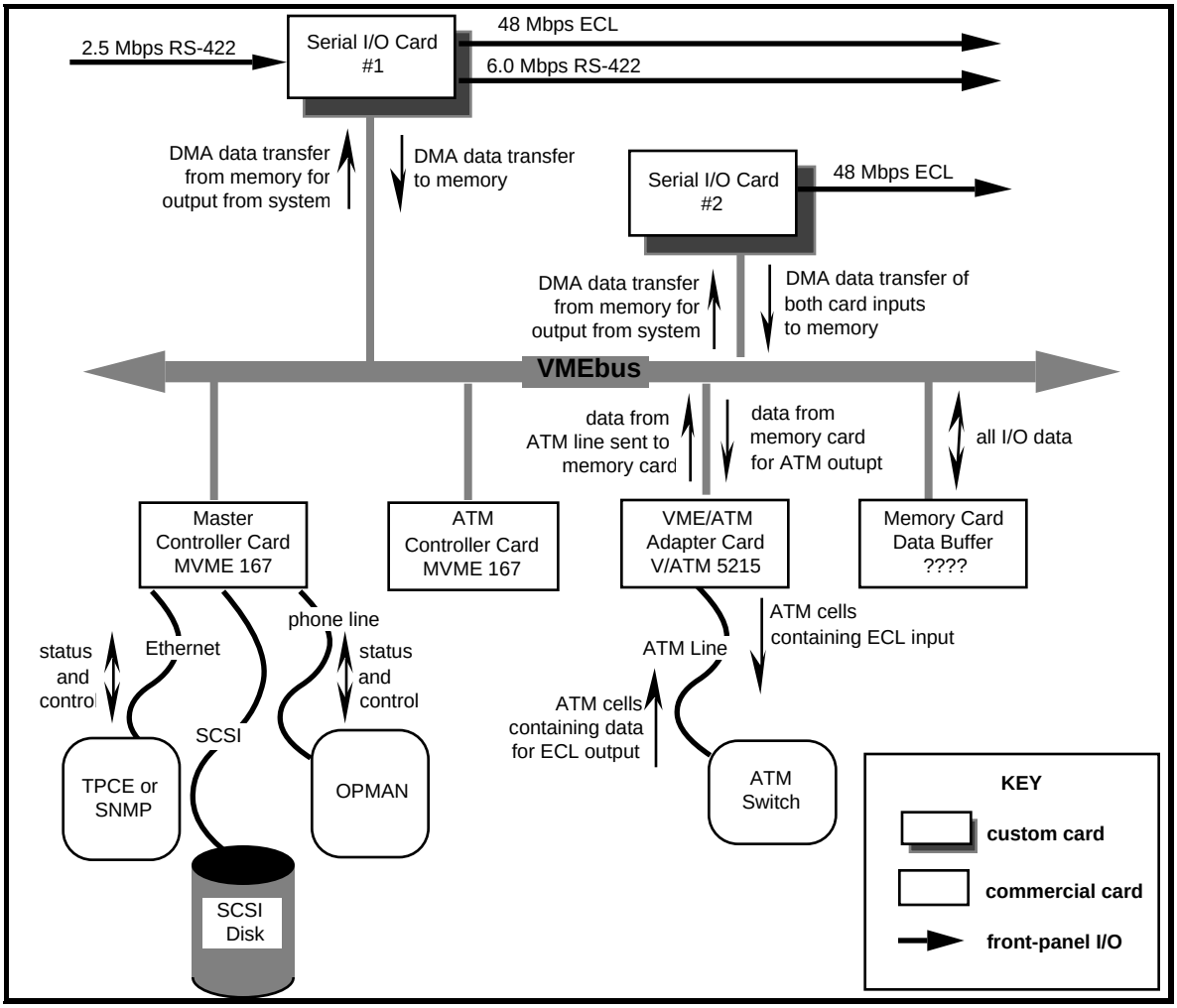


Figure 2-2. ATM Receive Mode Processing

2.3 FUNCTIONAL ELEMENTS

2.3.1 MCC

The MCC is the system central control and status point, but it never touches the data that the system processes. It serves as an interface between operator and system; it is responsible for providing setup and control to each of the subsystems and monitoring and reporting status of each subsystem. Any operator interface by which the system is controlled interfaces with the MCC, which disseminates the received setup information and/or reports back status gathered by polling the other subsystems.

Physically, this card resides in the left-most slot of the system backplane and it is a Motorola, MVME 167 32D card. If this card is reset (via front panel), then the whole system is reset. The MCC provides the dumb terminal interface for the local operator's interface, OPMAN, and it provides the Ethernet connection, which is used by remote operator interfaces. This card is also the interface to the SCSI disk.

2.3.2 MEMORY CARD

The MM6290 32M Memory Card is used for status and data transfer. It is a "dumb" card that does not include a processor. The memory card provides two types of data storage: zippy queues and status blocks.

The memory card address is defined by setting the card switches. For ACD System, the address range is 0x60000000 to 0x67F00000.

2.3.1.1 Zippy Queues

The Zippy Queue is the data transfer point between the SIO Card and the ATM Subsystem. It is a memory location where blocks of data are temporarily stored and then accessed.

The Zippy Queue is a MEDS-defined function. Two variables are fed into the function: block size and RAM name. The RAM name identifies the size and location of the memory available to the zippy function. The RAM is defined at boot-up in a file named "acd.sys," and the RAM is defined to reside within the Memory card's address range. The block size is defined by the operator with the SIO Card setup variable. Using the two variables, the zippy function places a sixteen byte header at the start of the RAM, and then divides the remaining RAM by the block size, which defines the number of blocks that can fit in the queue at any one time.

Zippy queues are used differently for transmit and receive modes. For transmit, there is one zippy queue for each SIO Card input channel. In transmit, the zippy queues contain data that is passed from the input channel and is available for ATM transmit. For receive, there are two zippy queues for each SIO Card output channel: each zippy queue holds addresses. One queue holds the address where data is stored, and the other queue holds addresses that are available for the ATM subsystem to use for data storage.

2.3.1.2 SIO Card Status Blocks

The SIO Card cannot be accessed by another processor during DMA transfers, which means that any time it is receiving or transmitting data, it can not be accessed. Since status that is available to the operator during data processing is obtained by the MCC periodically going out and polling status block locations, the SIO Card status block could not be located on itself.

The SIO Card status block resides on the Memory Card. The status block RAM is also defined at bootup in the "acd.sys" file. The SIO Card writes status to this location, and then the MCC can access it whenever needed.

2.3.3 SIO CARD

The SIO Card, which includes an onboard processor, has two modes of processing: input and output. However, before it can begin any processing, it must make some high-level decisions. According to operator setup, each active channel is identified, each channel's processing mode defined, and initial parameters downloaded.

2.3.3.1 Input Processing

Input processing is front-panel input that the SIO Card transfers to the zippy queue. Input is DMA'd to the zippy queue in the block size defined by setup. Input is either ECL or RS-422.

Input processing continues until either data input stops or the card receives a Shutdown or Disable command.

The SIO Card reports "Buffer Full" if a slot is not available in the zippy queue for data storage. If the SIO Card reports buffer full, data is dropped. This scenario indicates that the ATM subsystem is not reading data from the zippy fast enough, which means either data input is greater than system capabilities, the SIO Card and ATM subsystem are setup with conflicting block sizes, or the ATM driver has gone down and requires a hard reset. The latter is a rare occurrence and should not be immediately suspect.

During input processing, the SIO Card reports "Input FIFO empty" if the card's hardware input FIFO's do not see data. Two examples of errors that would cause this status follow:

a. The input clock into the card became so corrupted that it caused the FIFO flags to report incorrect status. In this case, data is corrupted, and to correct the problem the operator simply needs to issue a "Shutdown" and then an "Activate" command. This happens in a test scenario if the data source is powered off while it is still transmitting a clock signal to the SIO Card.

b. If the SIO Card is setup to expect the wrong input frequency, it can cause the card to report "Input FIFO empty" status. Most times, the card continues to process data correctly despite an incorrect setup. However, if the setup frequency and input clock are on opposite sides of 33 , it always causes periodic FIFO empty reports and blocks of data corruption.

2.3.3.2 Output Processing

Output processing is a more complicated function than input processing. Output processing is transfer of data from memory to the card's front panel for output from the system. Data is DMA'd from the memory to the card's output FIFOs. Output is either ECL or RS-422. Data is put into the zippy queue by the ATM subsystem, which must be receiving input via the ATM interface.

The operator sets up output processing for the expected raw data ATM input rate (NOT the ATM line cell rate, but the data rate after all overhead has been stripped). Data is output from the SIO Card at this rate, but the rate can vary by a delta value (also set up by operator). The nature of ATM transmission is bursty, which means that the ATM subsystems input to the SIO Card (via zippy queue) is bursty. To accommodate this, the SIO Card has to slightly vary its output rate to prevent either software buffers or hardware FIFOs from going empty or full.

The SIO Card monitors the number of blocks currently in the memory to determine if its current rate is either slightly too fast or slightly too slow. Output processing uses numerous operator-defined parameters to define how the memory block count status is monitored and how the output frequency is varied. Incorrect setup of these parameters can result in corrupted data processing. The setup parameters that determine output processing are as follows:

a. Output Frequency: Determines the baseline data output frequency. Any variations in frequency are calculated based on this initial value.

b. Sample Size: Determines how often the output processing checks the memory status and then adjusts output frequency based on the results. The sample size is measured in the number of blocks transferred. For example, if this value is set to 500, it means that after each 500 blocks transferred, the processing checks the zippy queue status. If the memory contains too few blocks, the output frequency is decreased. If the memory contains too many blocks, the output frequency is increased. A standard sample size for frequencies above 10 MHz is 5000 blocks.

c. Frequency Delta: Determine the number of Hz by which the frequency is either increased or decreased based on the sample results. A standard delta value for frequencies above 10 MHz is 20 Hz.

d. Minimum and maximum blocks: Determine the acceptable high and low block numbers in the memory. At sample time, if the number of blocks is greater than the maximum blocks, the frequency is increased. Conversely, if the number of blocks is less than the minimum blocks, the frequency is decreased. The maximum number of blocks memory can have is 263, and it should never get close to empty or full because the frequency will not be able to adjust quick enough to prevent an error condition. Standard values for frequencies above 10 MHz are 66 minimum blocks and 189 maximum blocks.

If the minimum block value is set too low, there is a risk that the memory will go empty and the SIO Card reports memory empty. If the maximum block value is set too high, there is a risk that the memory will overflow, and data will be dropped. In the latter scenario, the ATM subsystem receive status will report that the queue is full.

Ideally, the number of frequency adjustments should be infrequent, and if an adjustment occurs, the increment or decrement should be small. At the start (Activate command) of any data flow, the output processing fills the memory up to 130 blocks before it begins the queue status monitoring described above. This stabilizes startup and prevents the process from assuming the memory is empty at the start of a data flow and immediately decrementing frequency as a result.

If output processing is turned off (Shutdown command) while the ATM subsystem is still receiving data, and therefore still transferring data to the memory, a background task runs to read the data from the memory and dump it. This prevents the memory from being too full if the output processing is started back up, and it also prevents the ATM subsystem from reporting that the memory is full. However, when an activate command is reissued, there is a delay between when the background task is turned off and output processing is started back up, so the ATM subsystem could report several queue full status's.

2.3.4 ATM CONTROLLER AND V/ATM ADAPTER CARD (ATM SUBSYSTEM)

The ATM Subsystem is responsible for bi-directional data transfers between the zipply queue/memory and the ATM OC-3 lines. Two cards and their supporting software work in conjunction to provide the ATM subsystem function: the two cards are the ATM Controller and the V/ATM 5215. Like the MCC, the ATM Controller never touches the data. It sets up, controls, and monitors the V/ATM 5215 Adapter Card. The V/ATM 5215 is the hardware interface to the OC3 lines. It provides all AAL5, ATM Cell, and OC-3 packaging.

The specifics of the V/ATM 5215 (setup registers, available status, and buffers) are detailed in that card's supporting commercial documentation. The packaging functions are done by the commercial hardware/firmware. However, the card is setup and controlled by a driver that has been customized. Custom application software provides additional features to provide system-level setup, status, control, and data handling.

Like a demon, the ATM subsystem is always running. It can not be stopped and started (Activated and Shutdown), but its setup values can be changed. As soon as there is data available in the zippy queue for output onto OC-3 it is serviced, and as soon as data is input via OC-3 it is serviced. Because of this, the subsystem is always reporting status. For ATM transmit, if the subsystem does not find any data blocks in the memory, then it reports queue empty. For ATM receive and if data is being received, it counts the AAL5 packets even if the SIO Card output processing is not activated, and it may occasionally report queue full.

2.3.4.1 Receive ATM

For the receive ATM function, the ATM subsystem receives data from the OC-3 line. Based on the ATM cell VPI/VCI,data is routed for output. Data packaging is removed from the data, and the raw data is transferred to the memory. Prior to data transfer, the hardware verifies the AAL5 packet CRC and records any errors. If the cell's VPI/VCI does not match the setup, then the data is dropped.

If the ATM subsystem is ready to transfer data, but the memory is full, then data is dropped and this statistic is reported. Numerous input AAL5 packet-level counts and error conditions are also reported.

2.3.4.2 Transmit ATM

For the transmit ATM function, the ATM subsystem reads data from the zippy queue that has been put there by the SIO Card. The ATM subsystem packages the data into AAL5, ATM Cell, and OC-3 format. Each AAL5 packet includes a CRC value in the trailer. Each ATM Cell is stamped with a VPI/VCI value that is defined by system setup.

2.3.5 SCSI DISK MODULE

The SCSI disk module holds system application programs and setup files. The disk can not be remotely accessed unless it is mounted in the environment. It can be accessed locally from the vxWorks prompt.

2.4 SYSTEM-LEVEL I/O INTERFACES

2.4.1 DATA I/O

All data I/O to/from the ACD System is via the rear I/O panel. This includes serial ECL, RS-422, and OC-3 data I/O. Each rear I/O panel connector is cabled to the front panel of one card in an ACD Card Set.

2.4.2 OPERATOR INTERFACES

Three operator interfaces are available on the ACD System: OPMAN, SNMP, and card debug status screens. The operator can setup, control, and monitor the system via either the OPMAN or SNMP interfaces. The card debug status screens print card-level, application software debug statements. They are not a part of normal operating procedures, but (as the name suggests) are a feature that is available for a debug scenario.

2.5 NETWORK COMMUNICATIONS

2.5.1 ETHERNET

For the ACD System, the Ethernet interface provides setup and control functions, it does not provide telemetry data transfer.

The MCC card of each ACD card set provides the Ethernet interface. Unless the card set is configured to boot from a local disk, the Ethernet connection must be present for the system to even boot. Regardless of the boot source, the MCC must be connected to Ethernet for the SNMP interface to work.

For installation into a new site, the ACD System requires five IP addresses for each card set if it is booting off the network, which means that the system software is installed on a machine that is located on the network. If booting from local disk, the each card set requires two IP addresses.

If the MCC is connected to Ethernet, the operator can also remotely bring up OPMAN by using the "rlogin" command.

2.5.2 ATM

The ATM interface provides telemetry data I/O. It does not provide any system setup or control functions. The V/ATM 5212 card provides the ATM, OC-3 interface.

As stated, the ATM driver shall support AAL5 frames. The AAL5 frame format is illustrated in Figure 2-3 . The acronyms in the AAL5 trailer are defined as follows:

UU User-to-user information

CPI Common Part Indicator

LEN Length

CRC Cycle Redundancy Check

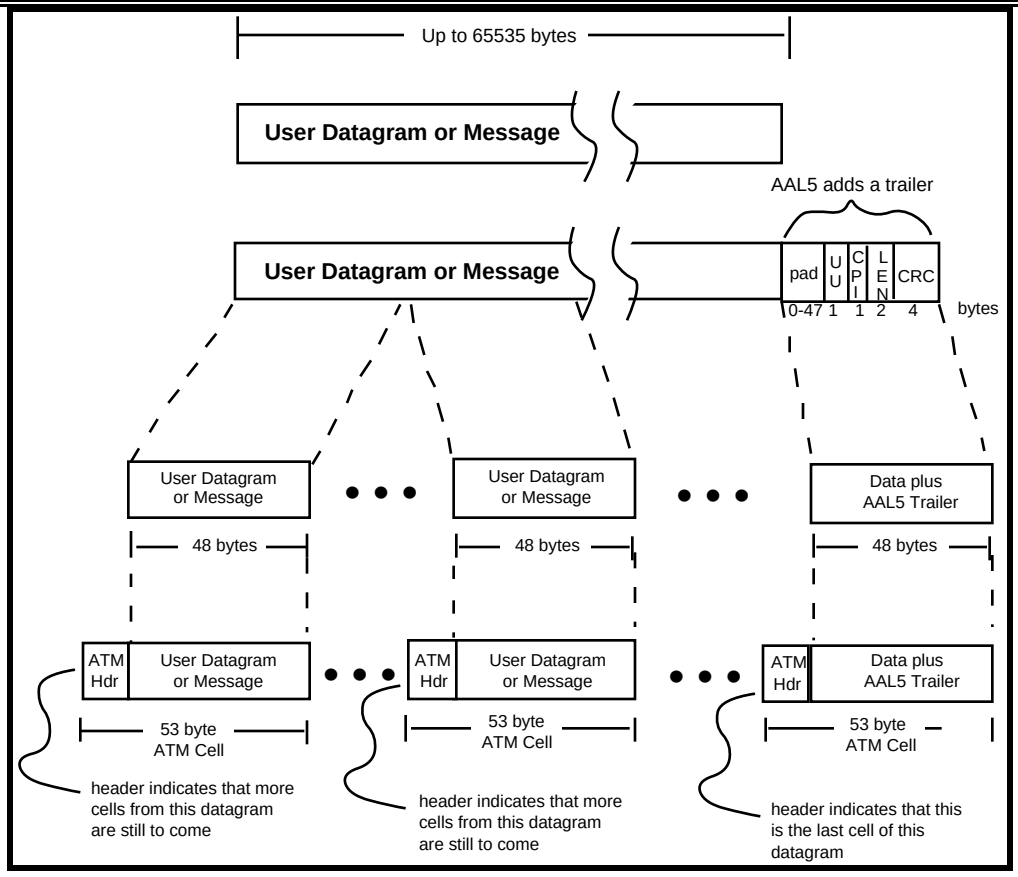


Figure 2-3. AAL5 Frame Format

The ATM cell format is 53 bytes: 5 bytes ATM header succeeded by 48 bytes of data, which is in AAL5 format. Figure illustrates the ATM cell format. The header fields are defined as follows:

GFC **The Generic Flow Control (GFC) is not yet standardized, but it allows multiple user's to share a line and guarantees the user the negotiated rate.**

VPI **The Virtual Path Identifier (VPI) works in conjunction with the Virtual Channel Identifier (VCI). They are both labels that are used for routing the cell. There are 256 virtual paths.**

VCI **The VCI is also a label used to route cells. There are 65,536 virtual channels for each of the 256 virtual paths.**

PTI **The 3-bit Payload Type Identifier (PTI) indicates whether the cell contains User or Network types of information and carries congestion notification. Bit one indicates if it is user data (1=user data); bit two indicates if the cell has encountered congestion (1=congestion); and bit three indicates if more cells are to come for that datagram or if that is the last cell of the datagram (1=last cell).**

CLP **The Cell Loss Priority (CLP) bit can be used to control congestion. If CLP is set to one, than that cell may be discarded under certain network conditions.**

HEC **The Header Error Control (HEC) reduces the likelihood of bit errors causing misdelivered information.**

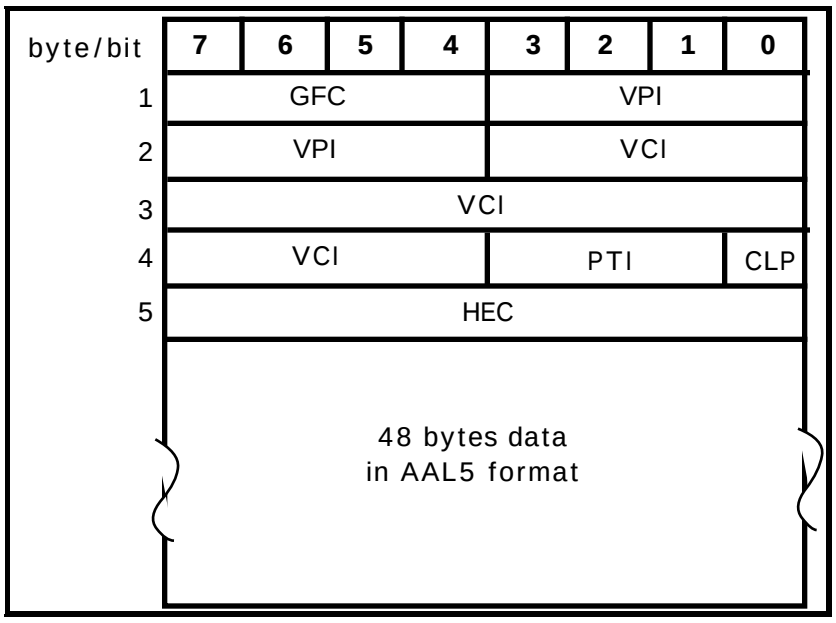


Figure 2-4. ATM Cell Format

SECTION 3
SYSTEM SOFTWARE

3.1 SOFTWARE ENVIRONMENT

Two software environments, VxWorks and MEDS, support the variety of commercial and custom hardware used in Code 521-designed systems. VxWorks provides a real-time operation environment; MEDS controls and monitors the system and the movement of data throughout the system. To control and monitor the system, SNMP software interfaces with MEDS.

3.1.1 VXWORKS

VxWorks is a high-performance, real-time operating system and a powerful development environment for real-time applications. VxWorks includes a fast and flexible run-time system, powerful testing and debugging facilitates, and a UNIX cross-development package (at the core of which lies VxWorks' extensive UNIX-compatible networking facilities).

The networking facilities allow VxWorks and UNIX to combine to form a complete, integrated development and operational environment; the UNIX system is used for software development and nonreal-time applications; VxWorks is used for testing debugging, and running real-time applications.

The VxWorks system can operate in a standalone manner, or can be networked with other systems running VxWorks or UNIX.

3.1.2 MEDS

MEDS is a real-time environment for card telemetry processing systems that provides a mechanism to control the system and report status. MEDS manages the mix of cards that are part of the ACD Card Set. MEDS works in conjunction with VxWorks to provide the system software platform upon which application-specific code is developed.

The overall MEDS software design is modularized into packages that supply general purpose system functions, such as operator interface support, status gathering support, command handling support, interprocessor communications, and network communications. Each package implements a set of functions that serve as a resource to application software, while hiding the details of their implementation. Consistent interfaces have been defined for each package, so code within a package can be changed without affecting the application code if the functions and interfaces remain the same.

Some MEDS packages exist as linkable libraries that the programmer uses to build a task, such as a command handler. Other packages exist as customizable source code files, where the applications programmer copies the source file, adds application-specific code, and compiles, for example, a status task. In all cases, the programmer does not need to know the details of MEDS implementation; only the interface to it. MEDS software is mainly written in C; assembly language is used for interrupt handlers.

A MEDS-based system is built by adding custom code to the general purpose MEDS code, which spares the application developer the burden of creating an infrastructure for each new system. It also adds consistency to system design, implementation, and maintenance. Figure 3-1 illustrates VxWorks/MEDS architecture.

3.2 SYSTEM-LEVEL SOFTWARE

The ACD system software controls each ACD card set. The ACD card set can either run from its local disk or run from software accessed via the Ethernet connection. Boot Parameters tell each system processor to look for software either on local SCSI or from a remote source. Regardless of the software's physical location, it is exactly the same software and the system operates the same.

The same directory structure is maintained regardless of the software's location. Only the top-level directories might be different to indicate the location of either the host's disk or the SCSI.

3.2.1 SYSTEM SOFTWARE ON LOCAL (SCSI) DISK

To boot and run from SCSI, the processor boot parameters must point to the SCSI disk, and the system software must be installed on the SCSI. Prior to software being installed to disk, the SCSI disk must be configured.

3.2.1.1 SCSI Disk Configuration

Configuration of the SCSI disk erases any software currently installed on the disk. The ACD System is delivered with a configured disk and software installed. Unless a new ACD card set and SCSI disk are being built up, an operator should never configure the SCSI disk.

Steps to SCSI disk configuration are as follows:

a. Change the MCC MVME 167 boot parameters so that it boots from acdscsi.cmd bootscript instead of acd.cmd bootscript. The new bootscript exports the SCSI disk.

b. Reboot the entire card set by pressing the MCC MVME 167 reset button.

c. At the vxWorks prompt, enter:

   scsiPhysDevIdGet

   This command returns a pointer to the specified SCSI_PHYS_DEV, or it returns NULL if the structure does no exist. An example return follows:

   value = 8245060 = 0x7dcf44

d. At the vxWorks prompt, enter:

   scsiBlkDevCreate(*address*)

   Address is the pointer returned at the last command. Given the value returned above, the command would be

   scsiBlkDevCreate(0x7dcf44)

   This command creates and initializes a BLK_DEV structure, which describes a logical partition on a SCSI physical block device. A logical partition is an array of contiguously addressed blocks; it can be completely described by the number of blocks and the address of the first block in the partition. This command either returns a pointer to the created BLK_DEV structure, or it returns NULL if there is an error. An example return follows:

   value = 8379064 = 0x7fdab8

e. At the vxWorks prompt, enter

dosFSMkfsOptionsSet(xf)

This routine sets the long filename option. Unless the option is invalid, it returns the following value:

value = 0 = 0x0

f. At the vxWorks prompt, type:

dosFsMkfs("/sd1",0x7fdab8)

This command initializes a device and creates a dosFs file system. It either returns a pointer to a dosFs volume descriptor, or it returns NULL if an error occurs. An example return value follows:

value = 8378900 = 0x7fda14

g. Reboot the entire card set by pressing the MCC MVME 167 reset button.

h. To check that SCSI configuration was completed without error, type the following command at the vxWorks prompt:

scsiShow

This routine displays the SCSI bus ID, logical unit number (LUN), vendor ID, product ID, firmware revision (rev.), device type, number of blocks, block size in bytes, and a pointer to the associated SCSI_PHYS_DEV structure for each physical SCSI device known to be attached to a specified SCSI controller.

3.2.1.2 SCSI Disk Software Upgrade

The SCSI disk uses the DOS operating environment. If a software upgrade is being made from the Unix environment, it is important to note that symbolic links can not be transferred. Anytime software is upgraded, it is strongly recommended that the old software first be moved to a new directory on the disk. Be certain that the software upgrade uses the established directory structure. If it doesn't, boot parameter will have to be changed to reflect the new paths to the software.

3.2.2 SYSTEM SOFTWARE ON NETWORK

To run from remote software, the ACD software must be installed on a host workstation that is connected to the ACD System via Ethernet. Boot parameters, which are programmed into PROMs that are installed on each processor board, point to the software location using a host Ethernet address and a known directory structure. Multiple systems can all point to the same software. The only complication with using the same software, is that any catalog setup change made from any system is saved to the same location.

To find the software, the boot parameters of each processor must be provided with the complete host path to the software location. The ACD has a set directory structure, but each host might have a unique top-level directory path that leads to the permanent ACD structure.

3.2.3 DIRECTORY STRUCTURE

The ACD directory structure starts at the directory named "devel." Beneath devel, the file locations and directory structure must not change or it affects system operation. The directories available beneath devel are as follows:

cat	all catalogs available through OPMAN
boot_scripts	contains the system- and subsystem-level bootscripts
cfg	
meds	contains the MEDS libraries and templates
opman	
rel_log	the release log is for information only. It provides information about the known errors and corrections/additions of each version of software that has been delivered. It also provides information about the software and hardware environment of each delivery.
snmp	this is the SNMP directory that contains the MIB, vxWorks get/set commands, the agent definitions, and the supporting code that allows SNMP to run.
subsys	contains the subsystem application code. It is divided into three subdirectories: atm, master, and sio.
tools	contains multi and rcs
vxworks	

3.2.4 SOFTWARE BOOT-UP

When the ACD System boots, each processor references boot parameters to find its software. The boot parameters point to the software location and include boot script file names. The boot script files include the step by step software initialization for each processor.

3.2.4.1 Boot Scripts

To find the boot script for a processor, use the path indicated by that processors boot parameters (refer to section 3.2.4.2). The boot script file names are listed below, and as a loose guideline, each can be found under the /devel/boot_scripts directory.

acd.cmd: assigns broadcast address, adds host name and address, loads custom set directory command, zeros memory card, loads OPMAN, calls "acd.sys" file, and loads SNMP deamon.

acd.sys: defines ram addresses and names and "ini" (subsystem initialization) files; sets path to catalog directory.

sio1Ram: start=60000000, end=602fffff, amode=a32
sio2Ram: start=60300000, end=6058000f, amode=a32
STATUS: start=60600000, end=606FFFFF, amode=a32
CDMAIL: start=60700000, end=607FFFFF, amode=a32

atm.cmd ATM Controller Card boot script. Loads application code and initializes software.

atm.sys: locates the page files for atm setup and status pages.

sio.cmd: SIO Card boot script. Loads application code, MEDS library.

acdbootline.cmd: holds the automatic SIO Card boot parameter configuration. A different function is used in this file depending on network (load_boot) or scsi (scsi_boot) bootup.

3.2.4.2 Network Boot Parameters

The boot parameters are the values that are programmed into battery backed up memory. To change these values, the operator must connect to each processor one at a time, and then reboot the system. Starting with the MCC, the operator watches the boot message on the processor and waits for the message that says "press any key to stop autoboot." After pressing return, the vxworks prompt appears on the screen. To change the parameters, enter "c" at the prompt. The first line of the boot script shows up. To change the setup, enter the new setup immediately after the current setup. Pressing return at each line of the boot parameters, allows the operator to scroll through the current setup. The current setup parameters can also be viewed by typing "p" at the vxworks prompt.

Many of the boot parameters change with the system's installation to reflect the host from which the system is booting and the IP addresses assigned to the system. Each ACD card set requires five IP address. One address for each of the following: MCC, backplane, SIO Card #1, SIO Card #2, and ATM subsystem. The five addresses must be a consecutive block of addresses, and the MCC card must be assigned the first address of the block. The backplane automatically assumes the second address of the block, and then addresses are assigned according to the processor number. For example, the MCC is processor number zero, which means it must be the first address of the block. The backplane automatically takes the second address. The SIO Card #1 is processor number one, so it must be the third address of the block; and the ATM subsystem is processor number two, so it must be the fourth address of the block.

The following list of boot parameters illustrates how boot parameters are assigned. All parameters that change with system installation are in plain text, and then a brief explanation follows. Parameters that are bolded, stay the same regardless of where the system is installed.

Example MCC boot parameters:

boot device	**:**	**ei**
processor number	**:**	**0**
host name	**:**	**vtroll --provide the host name on which the software is installed**
file name	**:**	**/23devel/acd/devel1.0/subsys/master/boot/vxWorks--provide the top-level path to the devel1.0 directory. From, the devel1.0 directory, the standard ACD directory structure is in place.**
inet on ethernet (e)	**:**	**128.183.97.64:ffff0000--provide the MCC IP address and broadcast for the network on which the system is installed.**
inet on backplane (b):		
host inet (h)	**:**	**28.183.97.20--provide the address for the host listed after "host name."**
gateway inet (g)	**:**	
user (u)	**:**	**vxd--provide the password**
ftp password (pw) (blank = use rsh):		
flags (f)	**:**	**0x20**
target name (tn)	**:**	**vxatmcs1--provide the name that matches the address listed after "inet on ethernet"**
startup script (s)	**:**	**/23devel/acd/devel1.0/boot_scripts/acd.cmd--provide the top-level path to the devel1.0 directory.**
other (o)	**:**	**vtroll=128.183.97.20:/23devel/acd/devel1.0--provide the host name and address and then the top-level path to the devel1.0 directory.**

Example SIO boot parameters:

boot device	:	sm=04000600
processor number	:	1
host name	:	vtroll--provide the host name on which the software is installed
file name	:	/23devel/acd/devel1.0/subsys/sio/boot/vxWorks--provide the top-level path to the devel1.0 directory. From, the devel1.0 directory, the standard ACD directory structure is in place.
inet on ethernet (e)	:	
inet on backplane (b)	:	128.183.97.66:ffff0000--provide the SIO Card IP address and broadcast for the network on which the system is installed.
host inet (h)	:	128.183.97.20--provide the address for the host listed after "host name."
gateway inet (g)	:	
user (u)	:	vxd--provide the password
ftp password (pw) (blank = use rsh):		
flags (f)	:	0x20
target name (tn)	:	vxatmcs1-p1--provide the name that matches the address listed after "inet on backplane"
startup script (s)	:	/23devel/acd/devel1.0/subsys/sio/boot_scripts/sio.cmd--provide the top-level path to the devel1.0 directory.
other (o)	:	(null)

Example ATM boot parameters:

boot device	:	sm=04000600
processor number	:	2-this number must change to three if there are two SIO Cards installed in the system
host name	:	vtroll
file name	:	/23devel/acd/devel1.0/subsys/atm/boot/vxWorks--provide the top-level path to the devel1.0 directory.
inet on ethernet (e)	:	128.183.97.67--provide the ATM Subsystem (MVME 167) IP address.
inet on backplane (b)	:	128.183.97.67:ffff0000--provide the ATM Subsystem (MVME 167) IP address and broadcast for the network on which the system is installed.
host inet (h)	:	128.183.97.20--provide the address for the host listed after "host name."
gateway inet (g)	:	
user (u)	:	vxd--provide the password
ftp password (pw) (blank = use rsh):		
flags (f)	:	0x0
target name (tn)	:	vxatmcs1-p2--provide the name that matches the address listed after "inet on backplane"
startup script (s)	:	/23devel/acd/devel1.0/subsys/atm/boot_scripts/atm.cmd--provide the top-level path to the devel1.0 directory.
other (o)		

3.2.4.3 SCSI Boot Parameters

To boot from disk, each processor's boot parameters are defined differently than they are to boot from network. Although, the system is booting from disk, it still requires five unique IP addresses. When the system boots from disk, the boot parameters work exactly the same way, but the parameters themselves need to reflect a different configuration.

The SIO Card always automatically configures its boot parameters, but it configures itself based on the MCC boot parameters. So the MCC must be set up correctly in order for the SIO Card to set itself up correctly.

The following information details the boot parameters for booting from SCSI:

Example MCC boot parameters:

boot device	**:**	**scsi=0,0**
processor number	**:**	**0**
host name	**:**	**vtroll --provide the host name on which the software is installed**
file name	**:**	**/23devel/acd/devel1.0/subsys/master/boot/vxWorks--provide the top-level path to the devel1.0 directory. From, the devel1.0 directory, the standard ACD directory structure is in place.**
inet on ethernet (e)	**:**	**128.183.97.124:ffff0000--provide the MCC IP address and broadcast mask for the network on which the system is installed.**
inet on backplane (b)	**:**	**128.183.97.125--provide the next IP address (sequentially)**
host inet (h)	**:**	**28.183.97.20--provide the address for the host listed after "host name."**
gateway inet (g)	**:**	
user (u)	**:**	**vxd--provide the password**
ftp password (pw) (blank = use rsh):		
flags (f)	**:**	**0x8**
target name (tn)	**:**	**vxatmcs1--provide the name that matches the address listed after "inet on ethernet"**
startup script (s)	**:**	**/23devel/acd/devel1.0/boot_scripts/acd.cmd--provide the top-level path to the devel1.0 directory.**
other (o)	**:**	**ei**

Example ATM boot parameters:

boot device	**:**	**sm=04000600**
processor number	**:**	**3--this number must change to 2 if there is only one SIO Card installed in the system**
host name	**:**	**vxatmcs1-this name must be the MCC target name**
file name	**:**	**/23devel/acd/devel1.0/subsys/atm/boot/vxWorks--provide the top-level path to the devel1.0 directory.**
inet on ethernet (e)	**:**	
inet on backplane (b)	**:**	**128.183.97.128:ffff0000--this address must be sequential from the MCC's address, but it is not an address on the network. Include the network mask. All addresses in the system are sequential starting with the MCC, and then maintaining the following order: backplane, SIO#1, SIO#2, ATM Controller.**
host inet (h)	**:**	**128.183.97.125--provide the system backplane address. This matches the MCC inet on backplane.**
gateway inet (g)	**:**	
user (u)	**:**	**vxd--provide the password**
ftp password (pw) (blank = use rsh):***--fill in the ftp password**		
flags (f)	**:**	**0x20**
target name (tn)	**:**	**vxatmcs1-p3--this is the host name and the processor number. It changes to "-p2" if there is only one SIO Card in the system.**
startup script (s)	**:**	**/23devel/acd/devel1.0/subsys/atm/boot_scripts/atm.cmd--provide the top-level path to the devel1.0 directory.**
other (o)		

3.3 APPLICATION-SPECIFIC SOFTWARE

3.3.1 SIO CARD APPLICATION SOFTWARE

The SIO card application software is called SIOV2.

3.3.1.1 SIOV2 Application Software Design

Figure 3-2 illustrates the SIO Card, its interaction with the system Master Controller, and the flow of data to and from the card. Data target and source are relative to the card, not to the system.

Three main tasks make up the SIOV2 software. Figure 3-3 illustrates the interaction between these tasks.

a. **Command Handler: The command handler is responsible for booting and initializing the SIO Card's hardware and for communication with the Master Controller. Once initialized, it listens for commands from the Master Controller in a continuous loop. The command handler may perform any combination of command-specific processing, writing control data to the global areas, and setting event flags.**

b. **Input Processing: The hardware automatically moves data from the data source to the input First-in, First-out (FIFO). The processor does not manipulate any data in the input data. The processor continuously issues DMA to transfer a block of data from the FIFO to the VMEbus data buffer. Once the data is in the FIFO, the data will be shifted out until the FIFO is emptied.**

c. **Output Processing: The output processor moves data from the VMEbus data buffer to the SIO output FIFO. Once data is in the output FIFO, the hardware will automatically pump out data to the ECL serial line. The processor does not manipulate any data in the output processing. The data level in the VMEbus data buffer is monitored by the CPU. The output data rata is adjusted accordingly to keep the SIO output FIFO from becoming empty. The processor also monitors the data level in the output FIFO, once it gets to almost empty flag, a DMA is issued to transfer a block of data from the VMEbus data buffer to FIFO.**

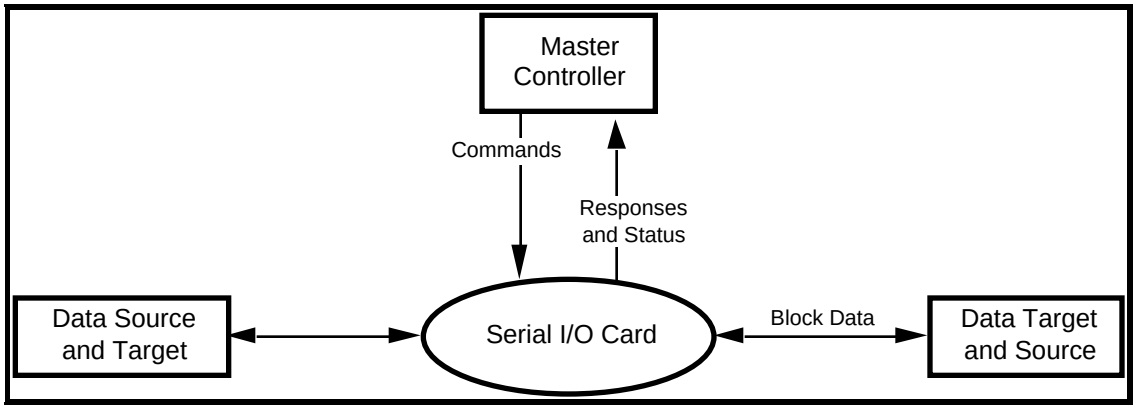


Figure 3-2. Context Model

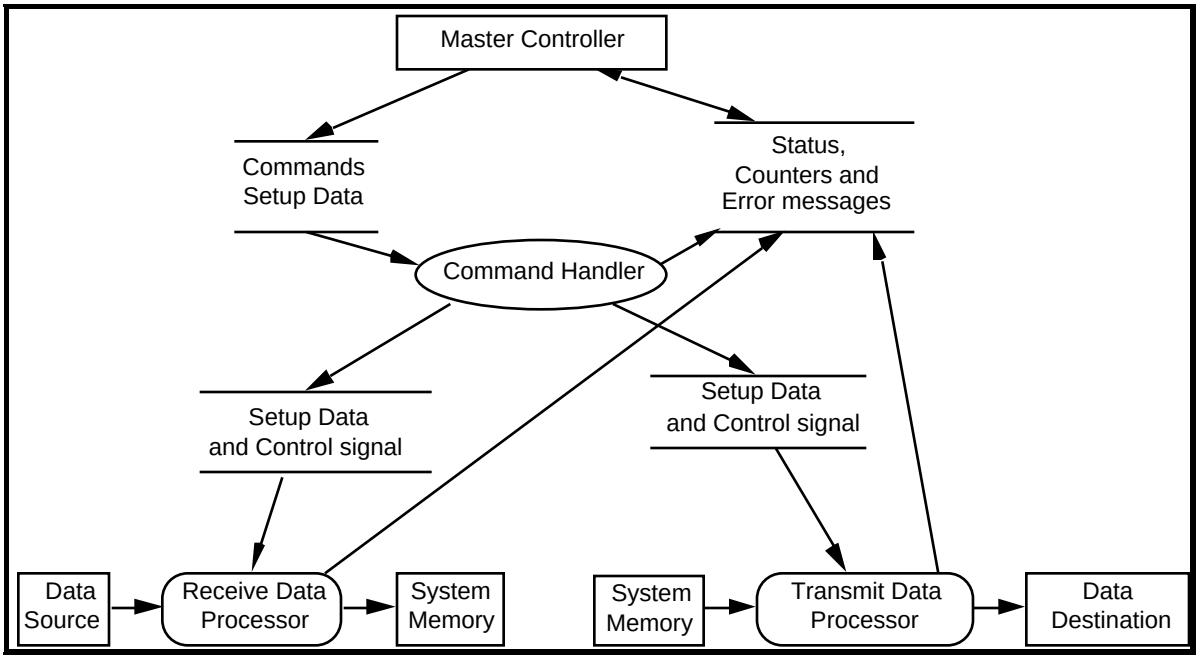


Figure 3-3. SIO Card Data Flow Diagram

3.3.1.2 SIOV2 Application Software Overview

The SIOV2 software is responsible for two main data processing functions: Input Processing and Output Processing. Additionally, the software must maintain communication with the system Master Controller Card, which is the interface to the operator. Figure 3-4 illustrates the two main SIOV2 telemetry processing functions.

Conceptually, the functions provided by the SIO software are simple. However, implementation of input and output processing requires some complicated software management. Several limitations/requirements complicate the SIO software:

a. There is one DMA engine per SIO Card, which means all active I/O tasks must share the resource and communicate availability to each other.

b. All input is multiplexed unto the same OC-3 line, the ACD System at the receiving end must be able to de-multiplex the data and output it as the original data stream and the original data rate. This requires record keeping on each DMA transfer.

c. Zippy queue communication requires one read task and one write task: multiple tasks can not use the same zippy.

d. Because the ATM input (SIO Card output processing) is bursty, the SIO Card NCO must be adjusted during output. Adjustments are made based on a current status: is the data transfer buffer filling up too fast or going empty too fast? To monitor a buffer status, record keeping for each stream of data must be separate from the other streams.

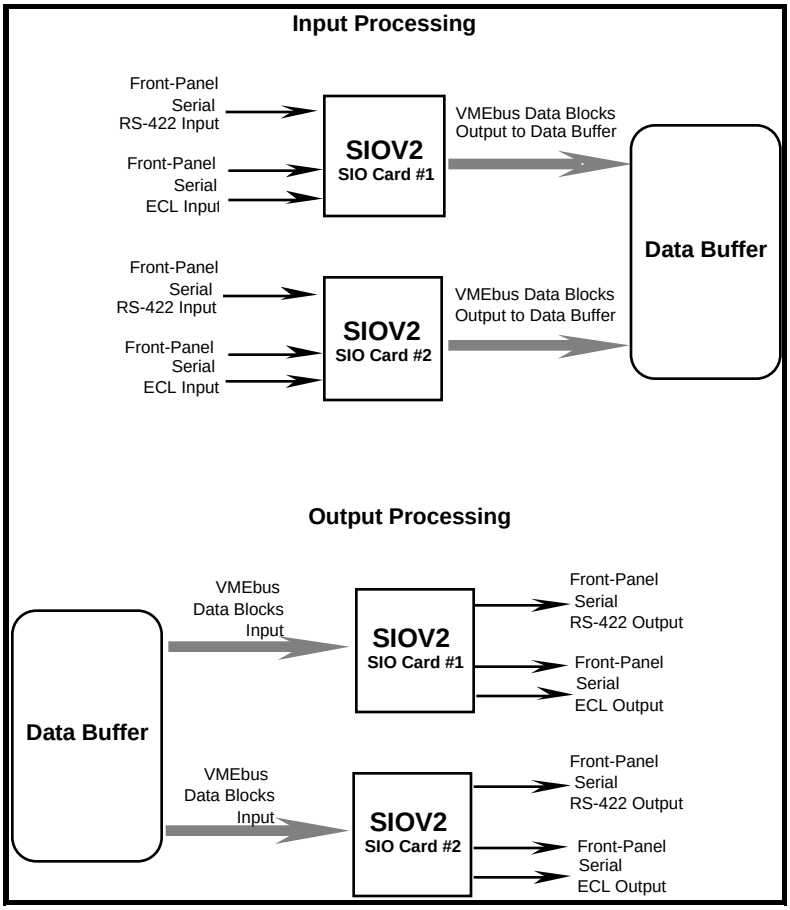


Figure 3-4. SIOV2 Input and Output

Since ATM protocol allows each cell to be stamped with a VPI/VCI value, each stream of data is tagged by assigning it an unique VPI/VCI value. Since the tagging occurs on the ATM subsystem side of processing, the burden of multiplexing/demultiplexing falls on the ATM subsystem. However, the SIO software must accommodate the ATM subsystem.

Figure 3-5 illustrates an overview of both SIO Input and Output processing software.

3.3.1.2.1 Input Processing

The SIO software Input Processing task is the more straightforward task. Each input channel on the card directly maps into one zippy queue. Any data received on a channel is always transferred to the same zippy queue. The only complication is that the DMA engine must be shared. Therefore, each channel's input task must wait for a semaphore to be released before it can perform a DMA transfer.

3.3.1.2.2 Output Processing

The SIO software Output Processing task is the more complicated task. Like the input tasks, an output task must wait for the DMA semaphore to be free before it can transfer data from its memory location to the output port.

The ATM subsystem can not DMA data to three separate locations. Therefore, all data, regardless of the cell VPI/VCI value, is DMA'd to the same location in memory. However, after the data DMA occurs, the address of each data block and its associated VPI/VCI value can be sent to multiple locations.

For each SIO Card, six zippy queues are set up: two queues per output channel. Each set of two queues consists of a "Data Address Zippy Queue," and a "Free Memory Address" zippy queue. Each output task (one task per active channel) interfaces with only one set of queues. The task reads an address from the "Data Address Zippy Queue," and this address points to the memory location where one block of data for that channel resides. The task DMA's the data from memory, and then releases that memory location for new ATM input by putting that address into the "Free Memory Address" Zippy Queue.

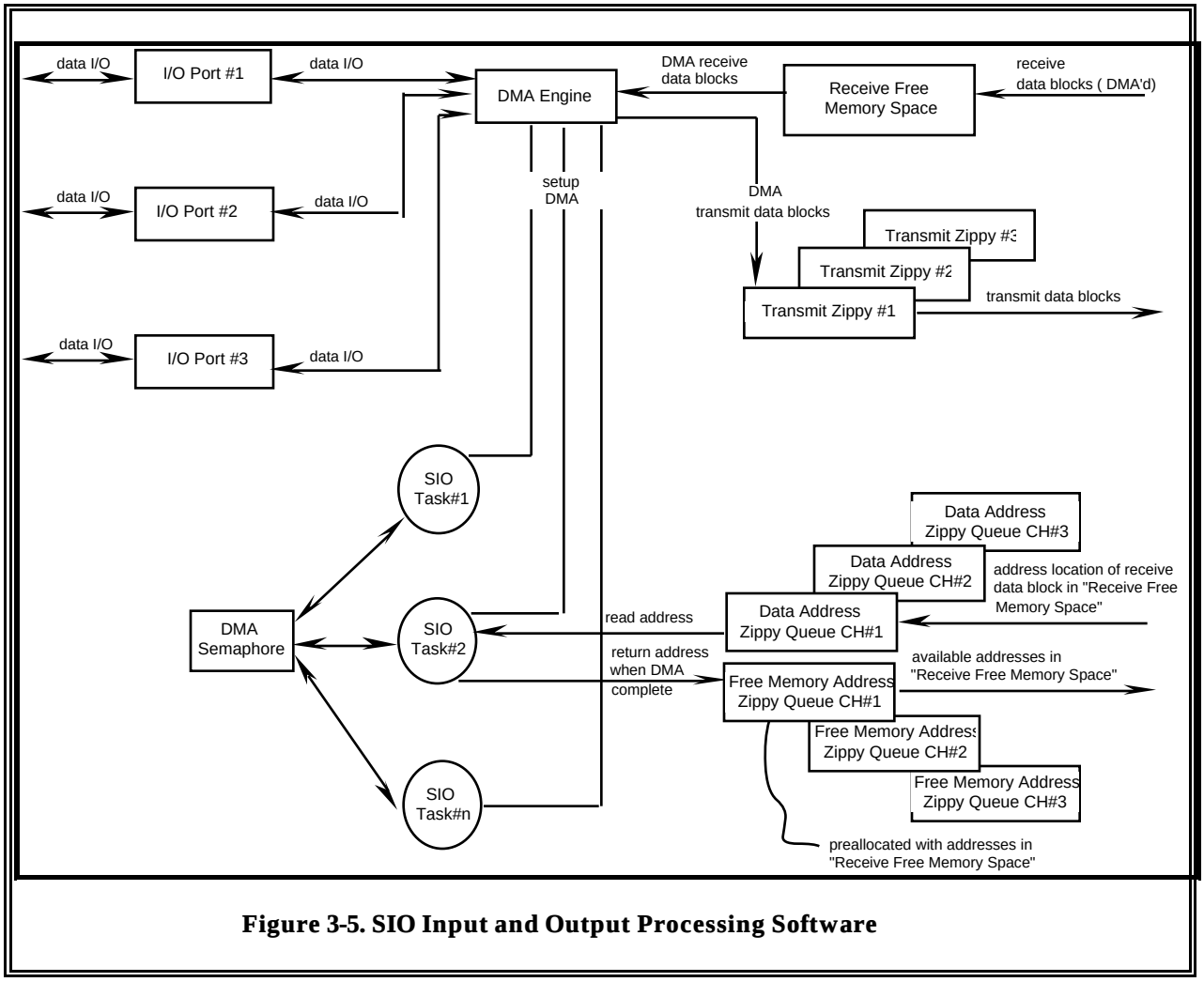


Figure 3-5. SIO Input and Output Processing Software

3.3.2 ATM APPLICATION CODE

To fit the ATM subsystem into the ACD System MEDS environment and to provide the system's functionality, an application layer of code was originally wrapped around the ATM driver. Originally, the ATM driver set up and controlled the V/ATM 5215 card. However, as system development continued, it became more efficient to integrate the driver and application code together. Therefore, a distinct and separate driver (from the application code) no longer exists. In addition to setting up and controlling the V/ATM 5215 Card, the application code interfaces setup and status to MEDS, passes data to and from the SIO Card, and it provides the necessary multiplexing/demultiplexing to transfer and reassemble multiple streams.

3.3.2.1 Transmit ATM

Figure 3-7 illustrates the ATM subsystem transmit mode. Functionally, the main application code task is the ATM Multiplexing task.

The SIO software passes all data to one of three zippy queues: each zippy queue represents the input of one SIO Card channel. The ATM Multiplexing task polls the three zippy queues checking for data input. Once data appears in a zippy, the task checks a table of VPI/VCI values that map to each zippy queue. The table is created from operator setup. Based on the table, the correct VPI/VCI is assigned to the ATM Cell header.

3.3.2.2 Receive ATM

Figure 3-6 illustrates the receive ATM function. The main application task is the ATM Demultiplex Task.

The complication on the receive side is that addresses must be assigned for DMA transfer to system memory even before the data arrives at the card (hardware definition). This makes it impossible to wait for ATM cell input, and then assign system memory destination based on cell header information. Instead, DMA addresses are preallocated 64 addresses at a time. As AAL5 data is received from the OC-3 line, it is sent one AAL5 frame at a time to a preallocated address. To prevent data loss, after data DMA, an available address must replace the used address in the list of preallocated 64 addresses.

At startup, the ATM subsystem creates six zippy queues per SIO Card and assigns one large memory space for data storage (receive side function only, additional zippy queues exist for transfer function). The zippy queues are divided into sets of two. Each set includes an "Available Memory Address" zippy and an "Address Location of Data" zippy. Each "Available Memory Address" zippy queue is preallocated with addresses that point to the data storage memory space.

At start of data flow, the first DMA from V/ATM 5215 goes to the first address in the preallocated 64 addresses. At DMA transfer, the VPI/VCI is stored with the corresponding data address. The ATM Demultiplexing task reads the data address and VPI/VCI, and then references the VCI value against a lookup table that results from operator setup. The table indicates which address zippy is transporting that VCI. Once the data address is passed to the correct zippy, the demultiplex task reads an address from the "Available Memory Address," and then replaces the used preallocated address on the receive ring with the address from the "Available Memory Address."

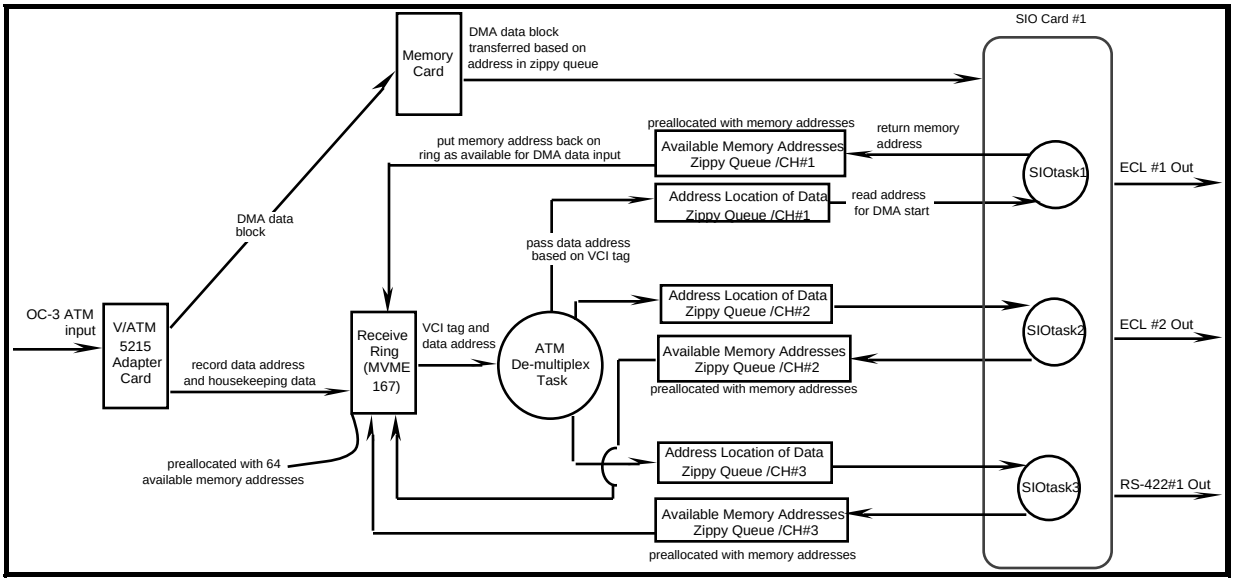


Figure 3-6. Receive ATM

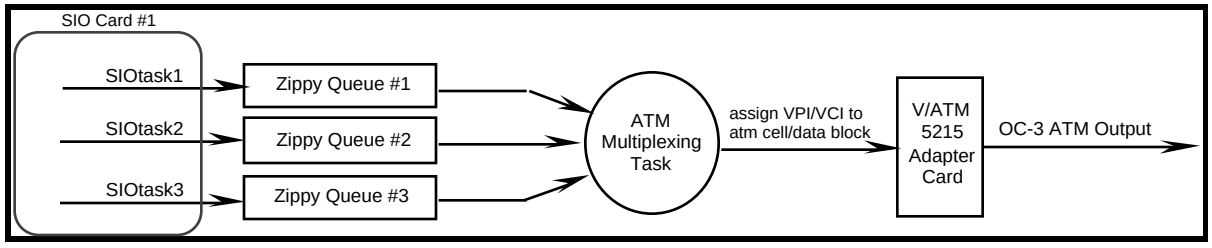


Figure 3-7. Transmit ATM

SECTION 4
SYSTEM HARDWARE

4.1 HARDWARE OVERVIEW

The ACD System is comprised of numerous hardware components that must be correctly connected and configured for the system to work. Since the complete system includes two power chassis and the one card chassis, to integrate the entire system it must be installed in a rack. The power chassis are rack mounted on top of each other, and then the card chassis is rack mounted above the power supplies. A 3U space should be left between the power chassis and the card chassis to allow for air flow. The power supplies must be cabled to the chassis back. Each end of each cable is clearly labeled to match the label of the connector to which it attaches.

System integration of the card chassis components is straight forward and fairly intuitive. Each card component must be configured, and then installed into the correct backplane slot. All I/O cabling is supplied with the chassis, the transition modules come with the chassis already connected to the correct slot, and the SCSI disk comes installed in the backplane and already cabled to the MCC. To connect a card component to the I/O panel, the cables provided with the chassis are connected to each card's front panel. The cables are clearly labeled to reflect the I/O connector to which it is connected.

4.2 UPS REQUIREMENT

The ACD Chassis power supply has a 15 A plug. It must be plugged into an UPS. If it is not plugged into an UPS, power glitches may result in unpredictable hardware failures.

4.3 COMPONENTS LIST AND MANUFACTURERS

Table 4-2 provides the hardware components that comprise the ACD System. The quantities listed comprise a fully populated system, which is two ACD Card Sets. Except for the Serial I/O card, all items listed are commercial products.

Most commercial products were purchased through Technico, located in Ivyland, PA. Technico can be reached via phone at (215) 957-9102.

Table 4-1

QTY	PART NUMBER	DESCRIPTION
1	TVS-6U8416-2-10M-R1001FT-8U-53126C	1) Technico 8U VMEbus chassis 2) 19" Rackmount chassis 3) Split 10 slot backplanes 4) 6U x 160mm card cage 5) Two, 10 slot J1/J2 monolithic backplanes 6) Onboard termination, auto jumper 7) Hinged front panel with acrylic window 8) Hinged rear panel 9) I/O panel to accommodate two ACD Card sets 10) Rear panel that accommodates connection to hot-swap, five channel, power subsystem through 1/4 20 studs and connector 11) QTY (4) Motorola MVME-167-032 CPUs. Two installed in each backplane: installed in first two slots of each backplane.
4	TVME-712-E2-M	Technico transition modules.
2	TDM2-2F1-1000H1-SC	Technico (Seagate) 1 Gigabyte hard drive and 2 megabyte floppy drive
1	N/A	Technico cable kit to connect two ACD card sets to Chassis I/O panel.
2	TRPS-02-10-G-5RO	Technico/HC Power Supply. Five channel, hot-swap power supply with the following rating: +5V @ 60 +12V @ 10 -5.2V @ 6 -2V @ 6 -12V @ 8
2	N/A	Technico cable harness from hot-swap power supply to rear of chassis.
2	Interphase V/ATM 5215	Interphase VMEbus to ATM adapter card.
2	Micro Memory MM 6290 D	Micro Memory memory board
2 to 4		Custom Serial I/O Card

4.4 COMMERCIAL COMPONENTS

4.4.1 OVERVIEW

Section 5 details commercial hardware configuration, and the operator should refer to the vendor's supporting documentation for design details of each commercial component. Except for the SIO Card, all hardware installed in the ACD System is a commercial product.

4.4.2 MVME 167

Two MVME 167 cards are included in each ACD Card Set. One operates as the MCC, and the second functions as the ATM Controller. Regardless of the function, the hardware for both cards is exactly the same. The card's function is defined by its slot location in the chassis, its address (burned into EPROM), and its master/slave configuration. The MCC should always be the left-most card in the backplane, and a card is set to the master or slave function with a single shunt jumper.

4.4.3 V/ATM 5215

One Interphase V/ATM 5215 card is included in each ACD Card Set. The card does not include a processor; therefore, it does not require an address in the system's memory map. It is accessed, setup, and controlled via the MVME 167 card that is functioning as the ATM Controller.

The card's hardware functionality is defined by the jumper locations. The card includes a motherboard and a daughterboard. It comes with the daughterboard attached, and it is installed in the system using the default setup with which it is delivered. As a precautionary step, all jumper locations should still be compared to the configuration information provided in Section 5.

4.4.4 MM 6290

One Micro Memory MM 6290 card is included in each ACD Card Set. This card does not include a processor, but it does require an address so that its memory can be accessed and used by the other cards in the system (which is the memory card's purpose). The cards address range is set with a series of switches that are turned on or off.

4.4.5 SCSI DISK

A delivered ACD System includes a SCSI disk that is already configured and loaded with operational software. If the disk is reconfigured or software is overwritten, all operational software is lost. SCSI disk configuration and software downloading is considered a part of system integration only and should not be performed by the end-user.

However, at system integration time, the disk does require configuration, and then the operational software must be written to the disk.

4.5 CUSTOM COMPONENTS

4.5.1 OVERVIEW

The only custom hardware component installed in each ACD Card Set is the SIO Card. The operator should refer to the SIO Card HDD for details of hardware design.

4.5.2 SERIAL I/O CARD

The SIO Card hardware design can be summarized by Figure 4-1. Details of the SIO Cards setup and control registers, as well as details of design, are provided in the SIO Card Hardware Definition Document.

The SIO Card includes a motherboard and a daughterboard (mezzanine). The mezzanine includes the card's 68030 processor. The SIO Card's address is burned into EPROM's on the mezzanine. The mezzanine has its own supporting documentation.

In addition to the address, each SIO Card has four sets of switches that must be set. The switches control ECL data I/O. Therefore, if these switches are not set correctly, the data I/O is corrupted.

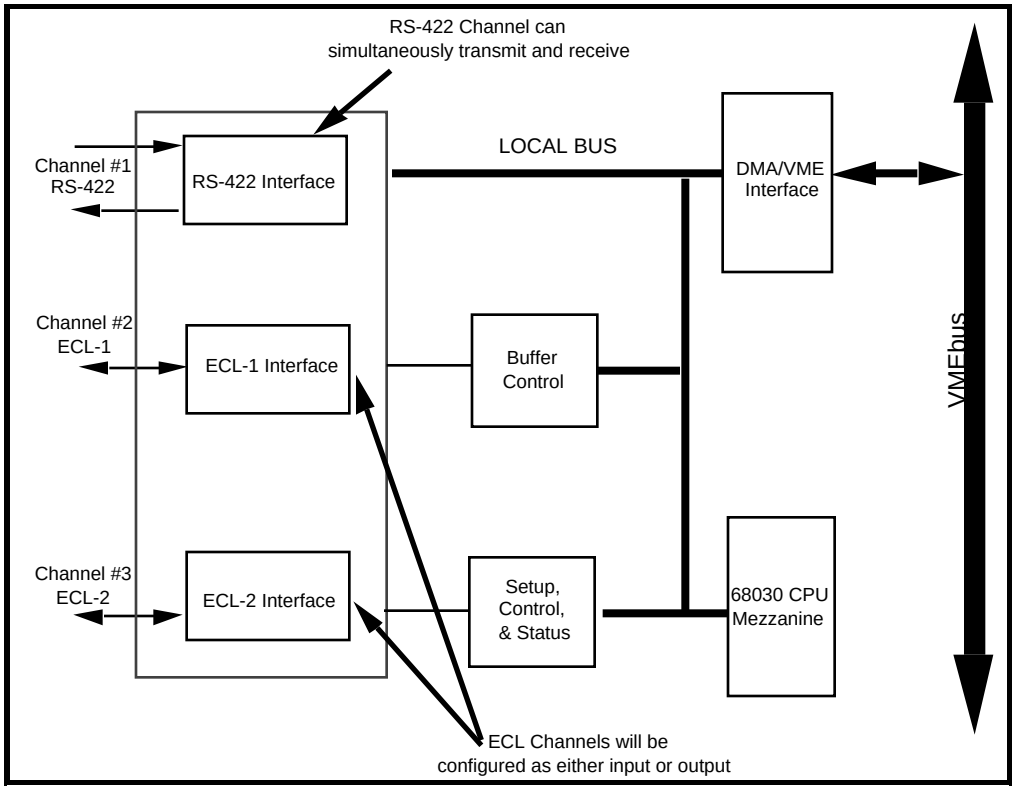


Figure 4-1. SIO Card Hardware Overview

SECTION 5
HARDWARE CONFIGURATION

5.1 CONFIGURATION OVERVIEW

The ACD System is delivered with cabling and components installed to reflect the configuration provided in this section. However, components can easily be removed, and cables can very easily be moved from one component to another. Minimally, the cabling of any new ACD should be verified against the information provided in this section. Front panel connections of the card chassis components must be correctly cabled to the I/O panel or system setup can be frustrating, if not impossible.

5.2 SYSTEM-LEVEL CONFIGURATION

Figure 5-1 reflects the location of system I/O panel and the cables that connect card chassis components to the I/O panel. The cables run across the top of the card chassis to the rear I/O panel. The cables can be accessed by removing the chassis top, which is connected with screws. However, this should not be necessary under normal operating circumstances. Each cable is clearly labeled with exactly the same label under a connector on the I/O panel. Connecting the cable to the card front panel, connects the card to the corresponding I/O panel connector.

Although not illustrated in Figure 5-1, the Ethernet connections are also available at the back of the chassis. Ethernet is attached to the MCC of a card set by connecting to the MCC transition module installed in the card chassis rear.

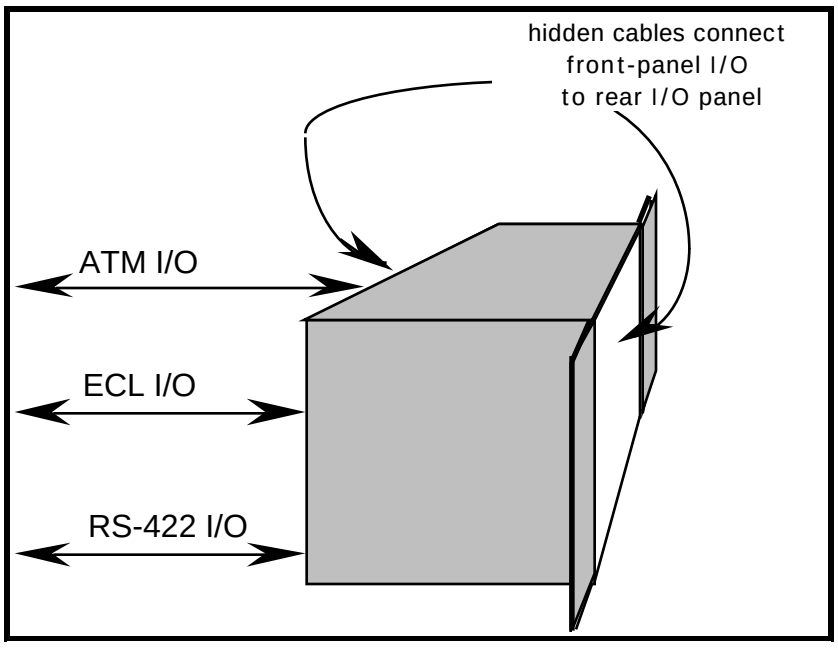


Figure 5-1. Data I/O

5.2.1 I/O PANEL AND TRANSITION MODULES

Looking at the chassis from the front, the left backplane is backplane #1, and the right backplane is #2. Looking at the back of the chassis, four transition modules are installed at the top-right. The two right-most transition modules connect to backplane #1: the right-most transition modules connects to the MCC and the other module connects to the ATM Controller Card (both are MVME 167s). Refer to Figure 5-2.

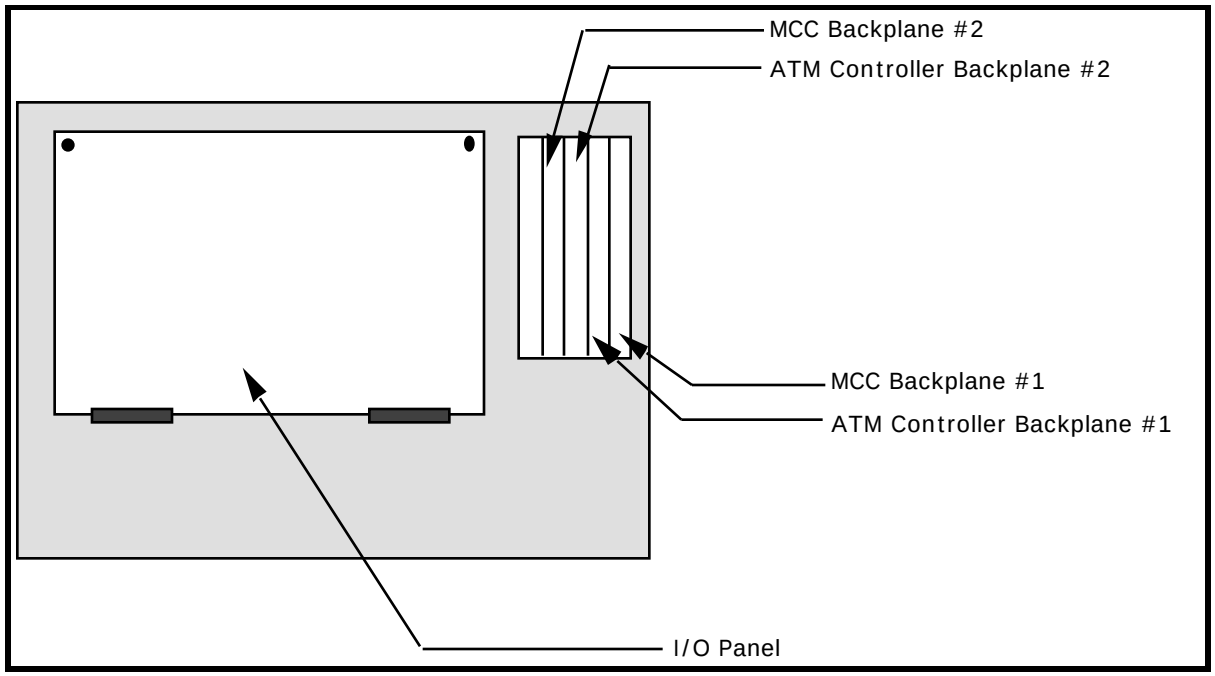


Figure 5-2. Rear Transition Modules

Figure 5-3 illustrates the ACD Chassis rear I/O panel. The I/O panel is designed to accommodate two ACD Card Sets installed in the same chassis (the chassis has a split backplane).

The following I/O connectors are assigned to backplane #1: ECL #1 to ECL #4, RS-422 #1, RS-422 #2, and ATM #1. The following I/O connectors are assigned to backplane #2: ECL #5 to ECL #8, RS-422 #3, RS-422 #4, and ATM #2.

For backplane #1, the I/O connectors are physically connected to the following components front panels:

a. ECL #1 to SIO Card slot #5 "ECL 1"

b. ECL #2 to SIO Card slot #5 "ECL 2"

c. ECL #3 to SIO Card slot #6 "ECL 1"

d. ECL #4 to SIO Card slot #6 "ECL 2"

e. RS-422 #1 to SIO Card slot #5 RS-422 DB-15 connector

f. RS-422 #2 to SIO Card slot #6 RS-422 DB-15 connector

g. ATM #1 to V/ATM Adapter board slot #4

For backplane #2, the I/O connectors are physically connected to the following components front panels:

a. ECL #5 to SIO Card slot #5 "ECL 1"

b. ECL #6 to SIO Card slot #5 "ECL 2"

c. ECL #7 to SIO Card slot #6 "ECL 1"

d. ECL #8 to SIO Card slot #6 "ECL 2"

e. RS-422 #3 to SIO Card slot #5 RS-422 DB-15 connector

f. RS-422 #4 to SIO Card slot #6 RS-422 DB-15 connector

g. ATM #2 to V/ATM Adapter board slot #4

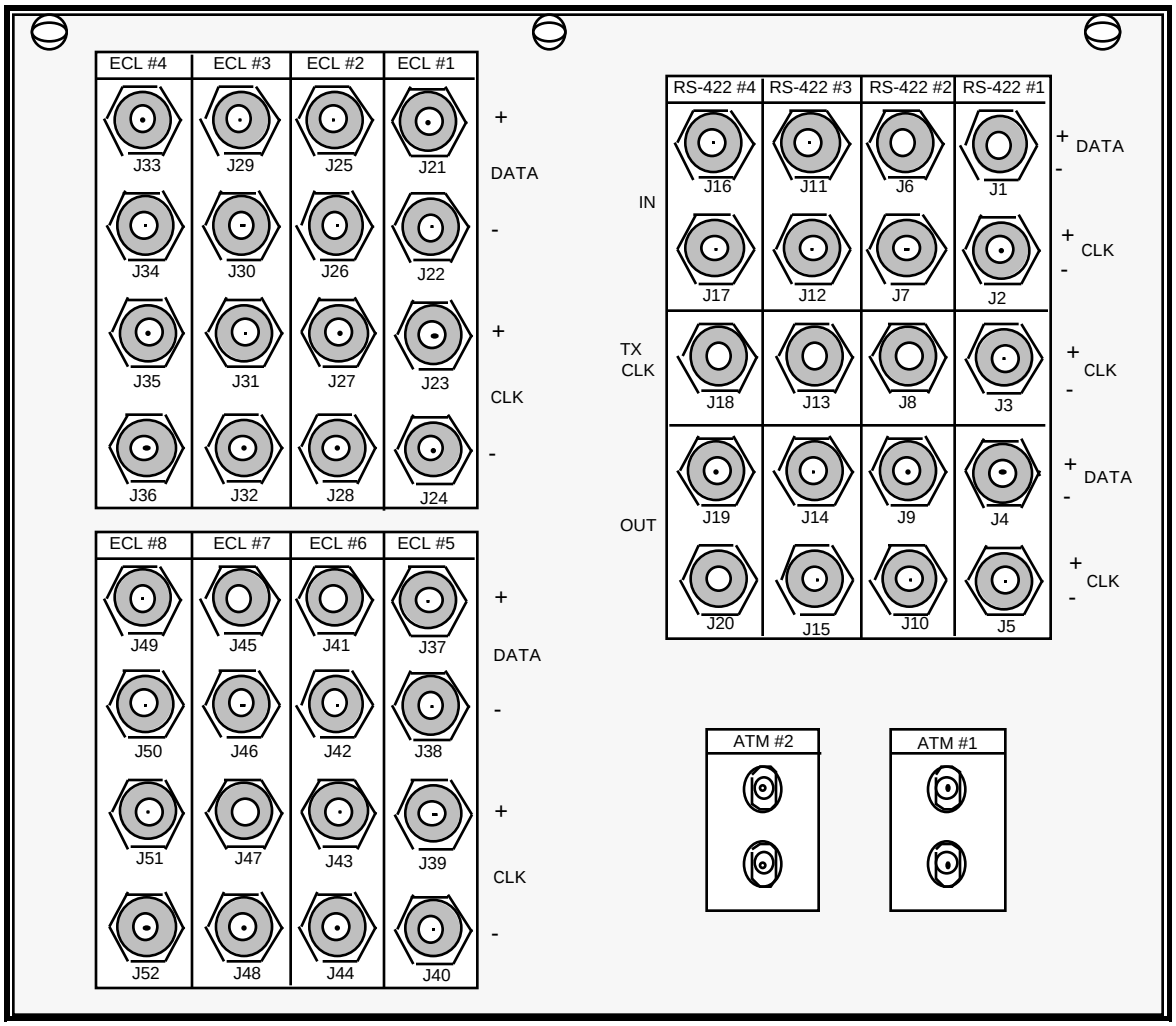


Figure 5-3. ACD Chassis Rear I/O Panel

5.2.2 ACD CARD SET POWER REQUIREMENTS

Table 5-1 lists the power requirements for each of the ACD Card Set components. The provided numbers are for only one backplane and they reflect peak values. For example, the disk module draws 3.3 A for a 5 second maximum.

Table 5-1. ACD Card Set Component Power Requirements

ACD Power Requirements for one set of cards:

COMPONENT	+12V	-12V	+5V	-5V	-2V
Serial I/O Card #1	--	--	5.0 A	1.5 A	1.5 A
Serial I/O Card #2	--	--	5.0 A	1.5 A	1.5 A
MM-6290	--	--	4.0 A	--	--
ATM Adapter Card	--	--	5.0 A	--	--
MCC MVME 167	1.0 Amax	1.0 Amax	4.5 A	--	--
ATM Controller MVME 167	1.0 Amax	1.0 Amax	4.5 A	--	--
Seagate Disk	3.3 A	--	1.5 A	--	--
P2 Adapter Board	--	--	1.0 A	--	--
Fans	2.0	--	--	--	--
TOTAL/1 card set	7.3A	2 A	32 A	3 A	3 A

5.2.3 SYSTEM MEMORY MAP

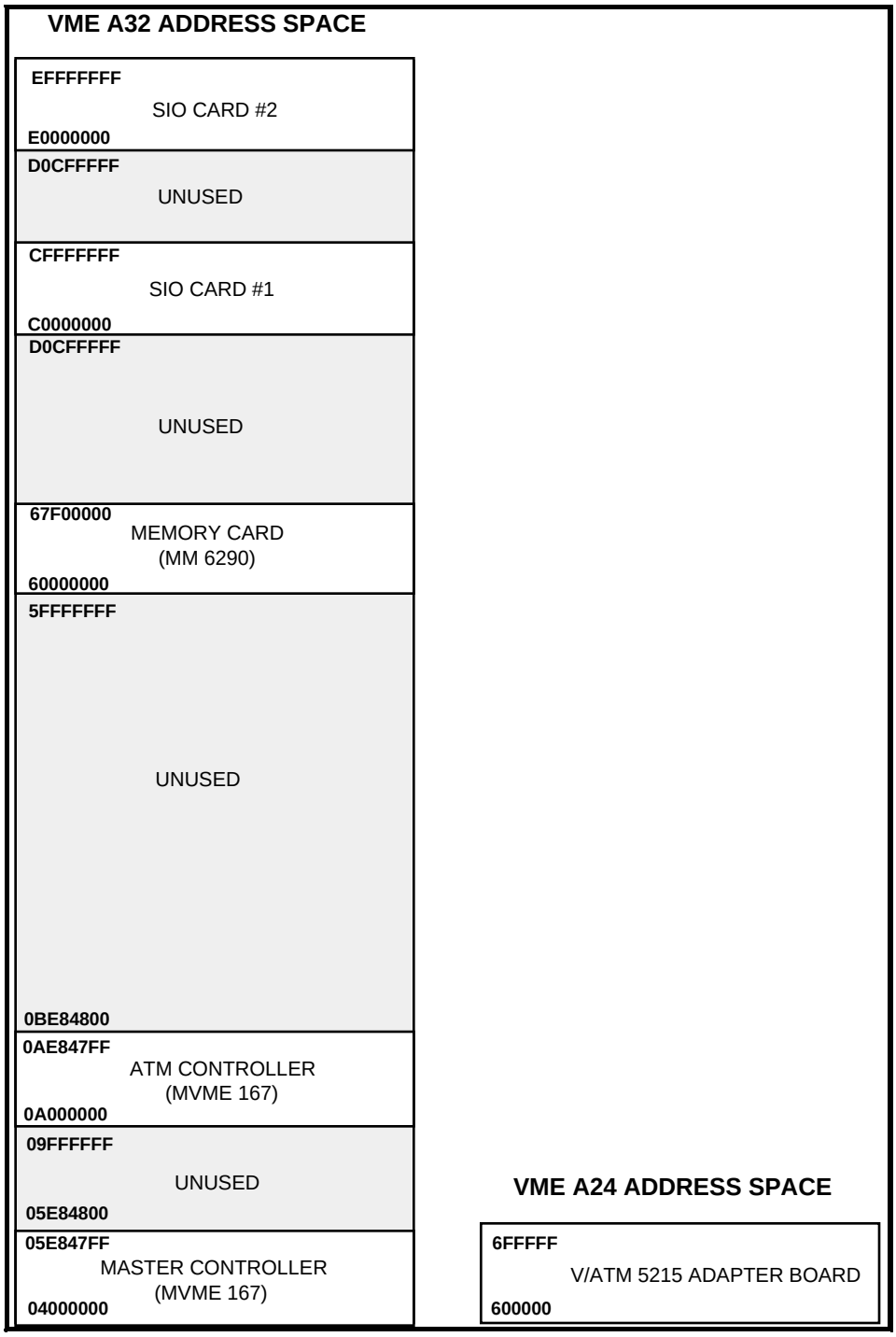


Figure 5-4. ACD Memory Map

5.2.4 VMEBUS PIN ASSIGNMENTS

Table 5-2 and Table 5-3 provide the VMEbus pin assignments.

Table 5-2. J1/P1 Connector VMEbus Pin Assignments

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DSO*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	AS*	GND	A18
21	IACKIN*	SERCLK	A17
22	IACKOUT*	SERDAT*	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12V	+5VSTDBY	+12V
32	+5V	+5V	+5V

*Refer to VME Specifications Book for signal descriptions.

Table 5-3. J2/P2 Connector VMEbus Pin Assignments

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	User-Defined	+5V	User-Defined
2	User-Defined	GND	User-Defined
3	User-Defined	Reserved	User-Defined
4	User-Defined	A24	User-Defined
5	User-Defined	A25	User-Defined
6	User-Defined	A26	User-Defined
7	User-Defined	A27	User-Defined
8	User-Defined	A28	User-Defined
9	User-Defined	A29	User-Defined
10	User-Defined	A30	User-Defined
11	User-Defined	A31	User-Defined
12	User-Defined	GND	User-Defined
13	User-Defined	+5V	User-Defined
14	User-Defined	D16	User-Defined
15	User-Defined	D17	User-Defined
16	User-Defined	D18	User-Defined
17	User-Defined	D19	User-Defined
18	User-Defined	D20	User-Defined
19	User-Defined	D21	User-Defined
20	User-Defined	D22	User-Defined
21	User-Defined	D23	User-Defined
22	User-Defined	GND	User-Defined
23	User-Defined	D24	User-Defined
24	User-Defined	D25	User-Defined
25	User-Defined	D26	User-Defined
26	User-Defined	D27	User-Defined
27	User-Defined	D28	User-Defined
28	User-Defined	D29	User-Defined
29	User-Defined	D30	User-Defined
30	User-Defined	D31	User-Defined
31	User-Defined	GND	User-Defined
32	User-Defined	+5 V	User-Defined

***Refer to VME Specifications Book for signal descriptions.**

5.3 CARD-LEVEL CONFIGURATION

5.3.1 COMMERCIAL CARDS

Each commercial card requires configuration prior to installation into the card chassis.

5.3.1.1 MCC MVME167

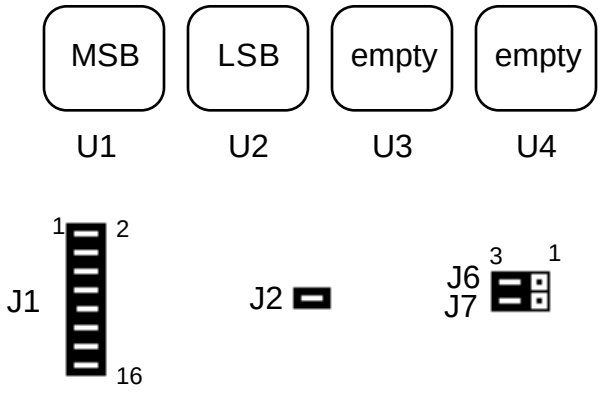
The MCC MVME 167 requires shunt jumpers installed in the correct location and EPROM's burned to reflect the correct VxWorks version and card address.

The MCC EPROM's have the following checksums:

MCC LSB checksum: **139321E**

MCC MSB checksum: **13A5F9A**

Total checksum: **27391B8**



5.3.1.2 ATM Controller MVME167

The ATM Controller MVME 167 requires shunt jumpers installed in the correct location and EPROM's burned to reflect the correct VxWorks version and card address. Shunt jumpers are installed the same as the MCC, except the J2 pin 1 to 2 shunt jumper is removed. Removal of this shunt jumper identifies the card as a slave instead of a master.

The ATM Controller EPROM's have the following checksums:

MCC LSB checksum: **16BD662**

MCC MSB checksum: **16BB065**

Total checksum: **2D786C7**

5.3.1.3 V/ATM Adapter Card

The V/ATM adapter card requires jumpers installed as illustrated in Figure 5-5.

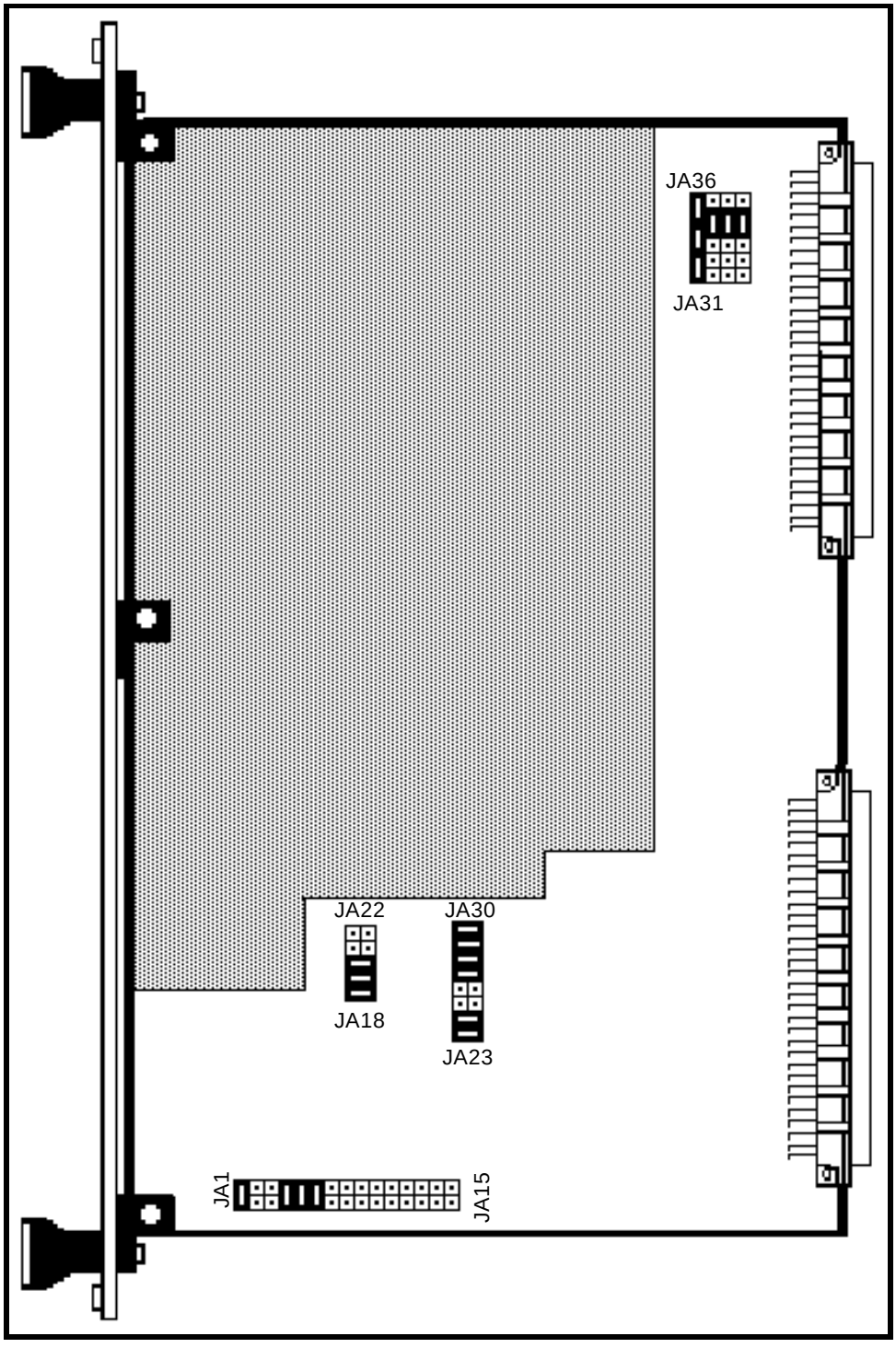


Figure 5-5. V/ATM Adapter Card Configuration

5.3.1.4 MM6290D Memory Card

The MM 6290 requires jumpers installed and switches set as illustrated in Figure 5-6.

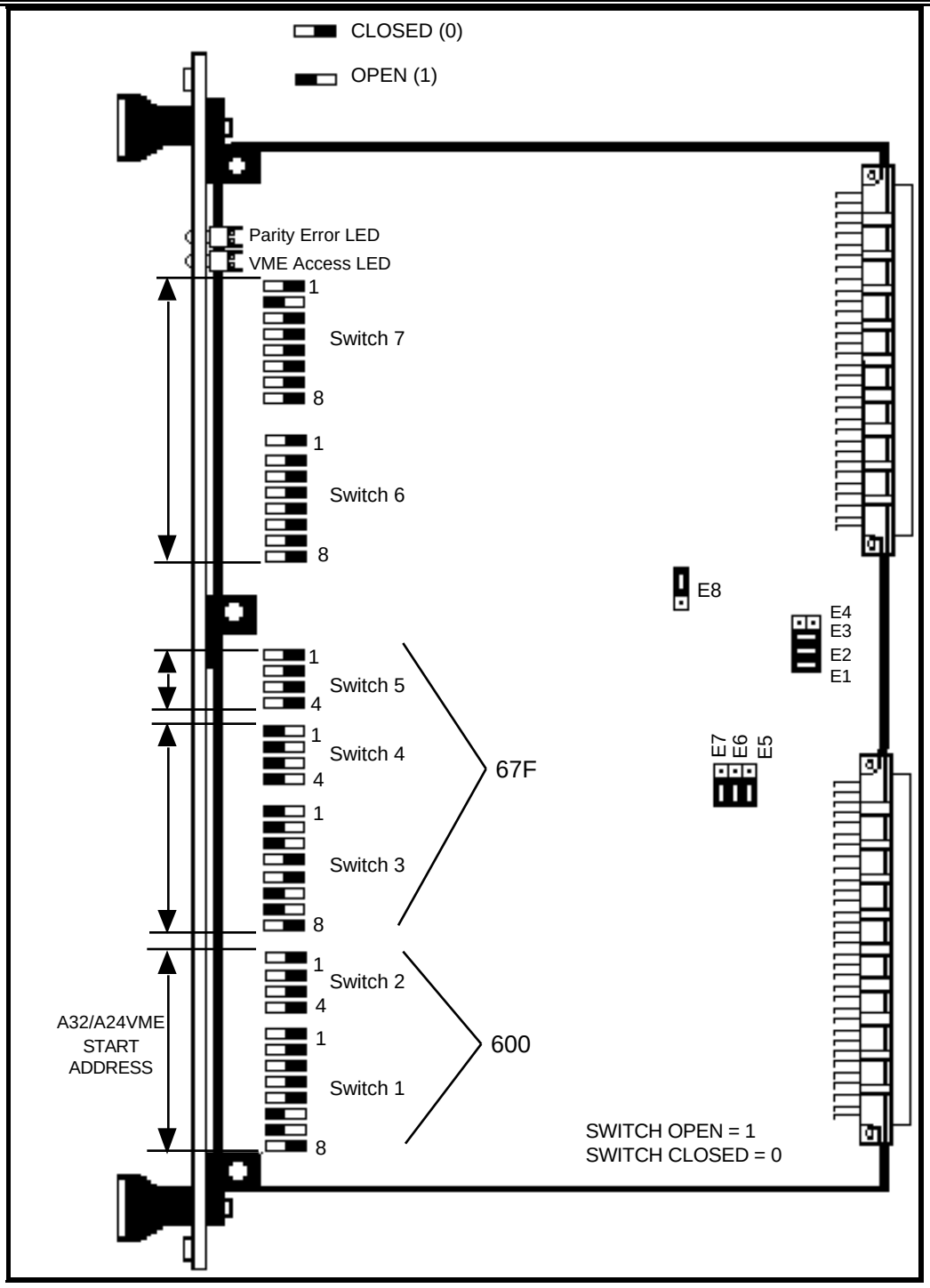


Figure 5-6. MM6290 D Memory Card Configuration

5.3.2 CUSTOM CARDS

Figure 5-7 illustrates the SIO Card. The switches are on the backside of the board. When the board is held with the front-panel correctly oriented (ECL connectors at bottom) and the back of the board showing, all switches are set completely counter clockwise. Parts U2, U5, U6, and U10 determine the card address and identify the card as either SIO #1 or SIO #2 in a card set. Address 0xC0000000 is SIO #1 and address 0xE0000000 is SIO #2.

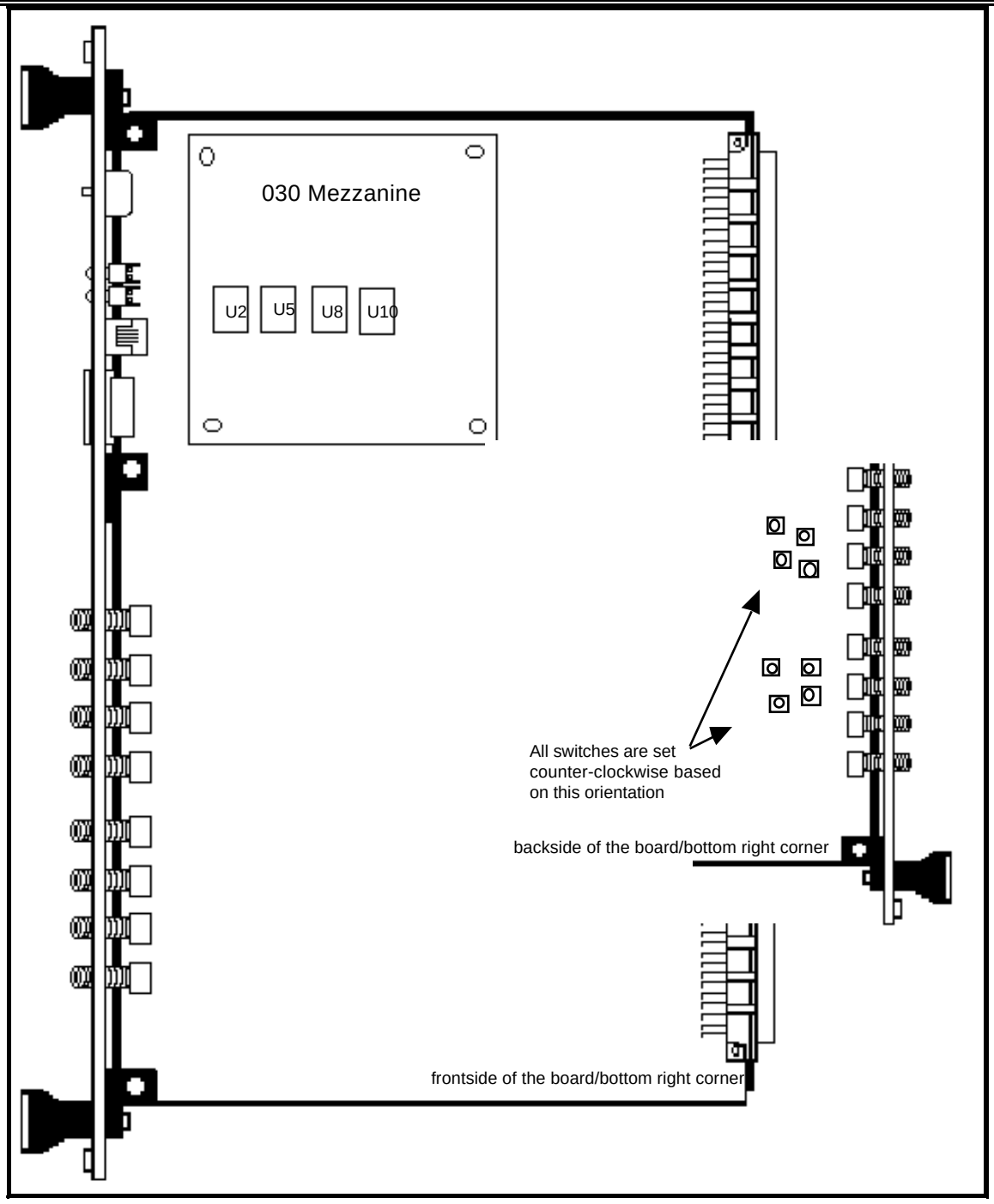


Figure 5-7. SIO Card Switch settings

5.4 CONTROLS, INDICATORS, AND FRONT-PANEL CONNECTIONS

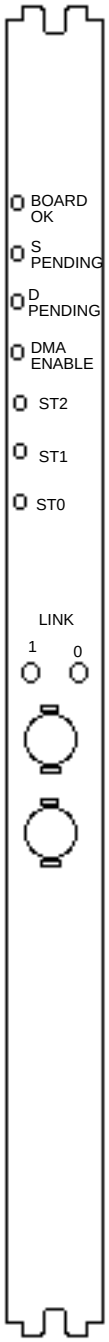
5.4.1 MVME 167-032B/32M- MCC AND ATM CONTROLLER CARDS

1. **FAIL LED (red): This LED lights when the BRDFAIL signal line is active.**
2. **STAT LED (yellow): This is the status LED. The MVME 167 decodes MC68040 status lines to drive this LED. A halt condition from the processor causes the LED to light.**
3. **RUN LED (green): When lit, this LED indicates that a local bus cycle is being executed.**
4. **SCON LED (green): When lit, this LED indicates that the card is the VMEbus system controller.**
5. **LAN LED (green): When lit, this LED indicates that the LAN chip is local bus master.**
6. **+12 LED (green): This LED lights when power is available to the transceiver interface.**
7. **SCSI LED (green): This LED lights when the card is the SCSI bus master.**
8. **VME LED (green): This LED lights when the card is using the VMEbus or when it is being accessed by the VMEbus.**
9. **ABORT Switch: This switch is not used by the ACD Card Set. If it were enabled by software, it would generate an interrupt at a user-programmable level that is normally used to abort program execution and return to the debugger.**
10. **Reset Switch: If the board is acting as system MCC, this switch resets the entire ACD Card Set. If the board is the ATM Controller, this switch resets only the controller.**

5.4.2 MM 6290 D/32M

1. **Parity LED (red): lights if parity error occurs during a read access.**
2. **VME Access LED (red): lights when card is accessed.**

5.4.3 V/ATM 5215



1. **BOARD OK:** Under normal operating conditions, this LED is green. This LED is red at power on, reset, or after a board failure. Use this LED in conjunction with ST0, ST1, and ST2 to further decipher the board's status.
2. **S PENDING:** When lit, this LED indicates that a host access to internal SRAM is in progress.
3. **D PENDING:** When lit, this LED indicates that a host access to internal DRAM is in progress.
4. **DMA ENABLE:**
5. **ST2, ST1, and ST0:** During card boot up and when the BOARD OK LED is green, these LED's blink, which indicates that the card's "Common Boot" interface is active. Once card boot is complete and BOARD OK LED is lit, these LED's cyclically blink up and down. When the BOARD OK LED is red, ST0 to ST2 LED's indicate the failure mode by reflecting the test number on which the card failed during self-test. Refer to Table for definitions of fail (BOARD OK LED is red) mode.
6. **LINK 1:** This LED indicates the optic link status. When the LED is lit, it indicates that the receive optics are detecting signals. At all other times, this LED is off. Refer to Table for additional information.
7. **LINK 0:** This LED indicates the card's daughterboard (front-end) status. If the LED is off, the reset has been deasserted, but the front-end is uninitialized. When the LED is blinking, the front-end has been initialized and is ready. Refer to Table for additional information.
8. **RX:** This is the card's OC-3 fiber receive connector. It should be connected to the transmit source.
9. **TX:** This is the card's OC-3 fiber transmit connector. It should be connected to the receive destination.

Table 5-4. ST0, ST1, and ST2 Definitions in Failure Mode

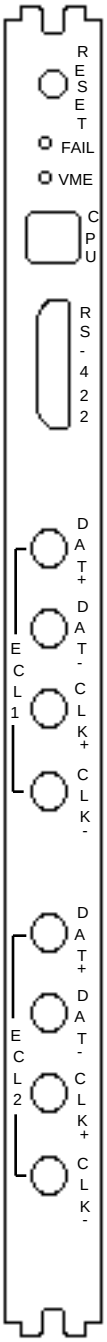
"BOARD OK" LED is Red				
ST2	**ST1**	**ST0**	**"on" LED's are blinking/ CBII status**	**"on" LED's are NON-blinking/HARI status**
off	off	off	reserved	reserved
off	off	on	MEM diagnostics failed	reserved
off	on	off	DMA diagnostics failed	reserved
off	on	on	FE diagnostics failed	reserved
on	off	off	reserved	F/W Panic
on	off	on	reserved	VME Bus Error
on	on	off	reserved	VME Bus Timeout
on	on	on	Power on, Reset	Power on, Reset

***CBII and HARI are modules on the V/ATM Card. Refer to that card's documentation for details.**

Table 5-5. LINK0 and LINK1 LED Status

LED state	Optic Link Status
LINK 0 is off	Fiber Optic link is down
LINK 0 is green	Fiber Optic link is "ok"
LINK 1 is yellow	Power on, Reset
LINK 1 is off	Power on complete
LINK 1 is blinking yellow	Daughterboard is initialized and running

5.4.4 SIO CARD



1. **RESET button:** **This button performs a hardware, card-level reset. It is mainly used for card-level debug prior to installation in an ACD Card Set. For ACD Card Set problems the MCC reset button should be used, not the SIO Card reset button.**
2. **FAIL**
3. **VME**
4. **CPU connector:** **This "phone-type" connector interfaces the card CPU to a dumb terminal. The SIO Card subsystem software reports software status and setup messages to this port. The information available through this port is developmental in nature. In extreme cases of system-level debug, this information may need to be accessed, but normal operating procedures should never require that this port be accessed.**
5. **RS-422 connector:** **This connector provides RS-422 data I/O to and from the card.**
6. **ECL1/DAT+:** **This connector provides ECL Channel number one positive data I/O.**
7. **ECL1/DAT-:** **This connector provides ECL Channel number one negative data I/O.**
8. **ECL1/CLK+:** **This connector provides ECL Channel number one positive clock I/O.**
9. **ECL1/CLK-:** **This connector provides ECL Channel number one negative clock I/O.**
10. **ECL2/DAT+:** **This connector provides ECL Channel number two positive data I/O.**
11. **ECL2/DAT-:** **This connector provides ECL Channel number two negative data I/O.**
12. **ECL2/CLK+:** **This connector provides ECL Channel number two positive clock I/O.**
13. **ECL2/CLK-:** **This connector provides ECL Channel number two negative clock I/O.**

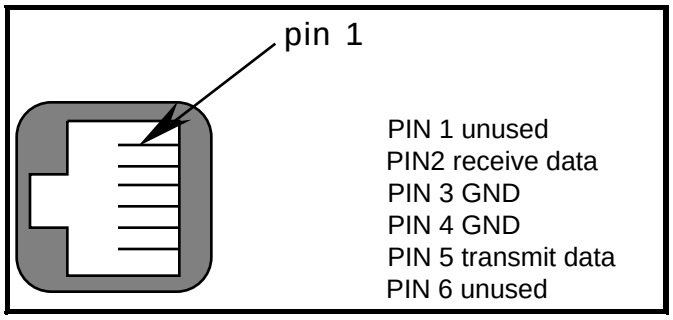


Figure 5-8. SIO Card Front-Panel CPU (RJ-11) Connector Pinout

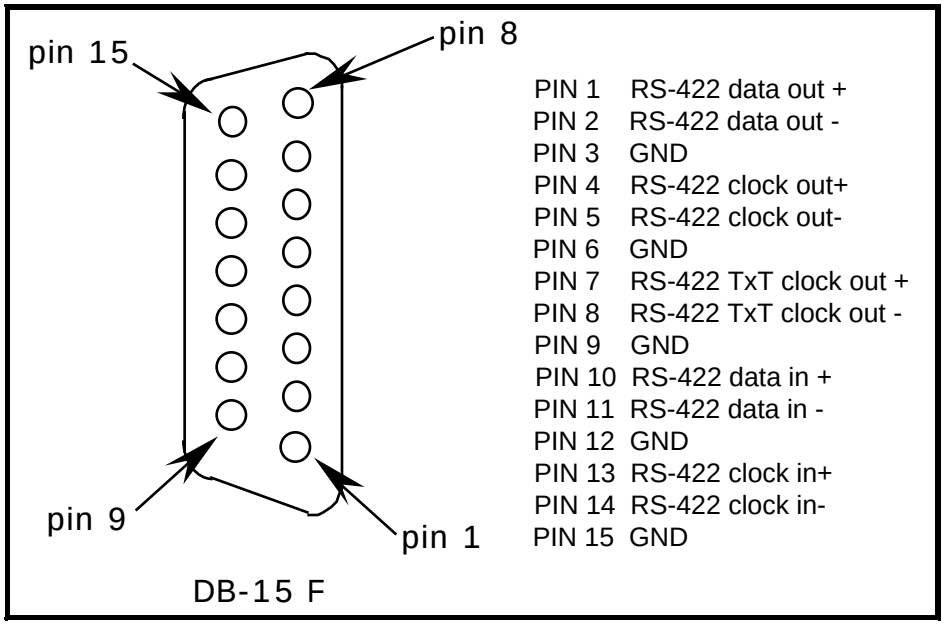


Figure 5-9. SIO Card RS-422 Front-Panel Connector Pinout

SECTION 6
OPERATOR INTERFACE

6.1 OVERVIEW

The ACD System is delivered with a local operator interface, OPMAN, and a SNMP MIB II. Each card set is run from a separate OPMAN interface.

6.2 OPERATOR INTERFACE CONNECTIONS

Figure 6-1 illustrates the physical interfaces for the OPMAN and SNMP operator interfaces.

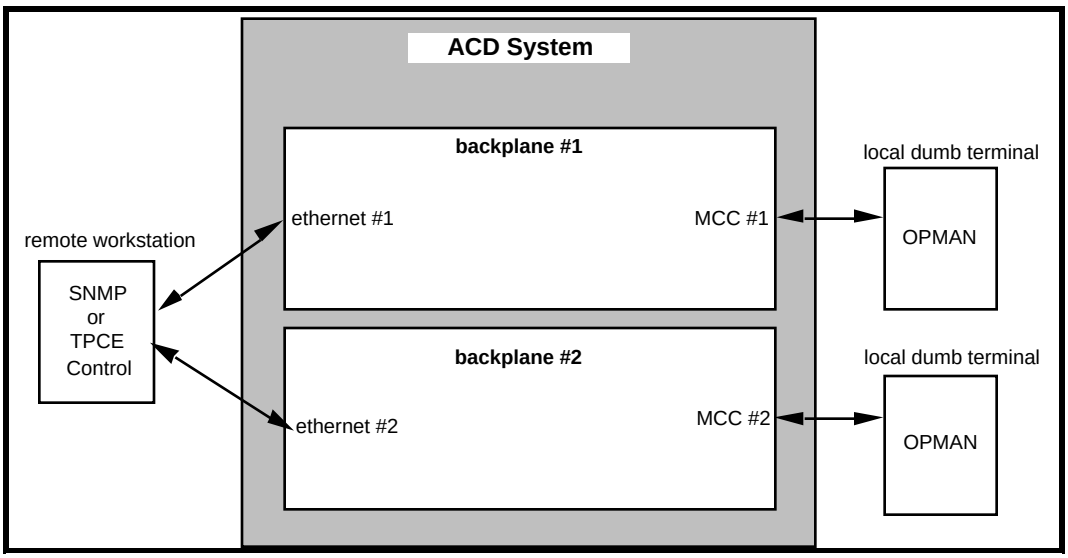


Figure 6-1. ACD System Operator Interfaces

6.3 OPERATOR INTERFACE PROCEDURES

Using OPMAN is detailed in the ACD User's Guide. The user's guide also maps SNMP commands to the equivalent OPMAN command.

6.3.1 SYSTEM SETUP

ACD setup is based on files called catalogs. The operator defines a catalog and assigns it a name. To define a catalog, the operator uses an edit function to pick the subsystems to include (SIO#1, SIO#2, and ATM). To provide data I/O to or from an OC-3 line, the ATM subsystem must always be present in a catalog. However, if two SIO Cards are included in a card set, the operator only needs one card included in a catalog. Two cards can be included in a catalog to provide numerous data channels, but two cards are necessary for the system to function.

Once the subsystems are included in a catalog, each subsystem requires detailed setup information be defined in the catalog. To allow definition of the detailed subsystem setup, OPMAN provides one screen with numerous pop-up windows per subsystem. The details of each setup screen are provided in the user's guide.

Once subsystems are chosen for setup and the detailed setup information defined, the operator saves the file with a unique name. Once the file is saved, that setup information contained in the file can be downloaded to the system at any time by identifying that file by name.

6.3.2 SYSTEM CONTROL

Once catalogs are created, several top level commands allow the operator to control the ACD Card Set. The basic set of control commands are: LOAD, ENABLE, ACTIVATE, and SHUTDOWN.

The LOAD command initiates a request for a catalog name. When the catalog name is entered, the setup information contained in that file is loaded into each subsystem.

The ENABLE command turns each subsystem on according to the information currently loaded into each subsystem. Data processing does not start until and ENABLE is issued.

ACTIVATE combines LOAD and ENABLE. It reduces the number of steps required to start data processing, but does not allow the operator to examine the setup information loaded prior to starting the system.

To stop data processing, the operator issues a SHUTDOWN command.

6.3.3 SYSTEM STATUS

Each subsystem has a numerous pieces of status information that it reports back to the MCC. OPMAN provides one screen per subsystem to illustrate the current status of each subsystem. It also provides one summary page that provides a top-level look at each subsystem's current status.

6.4 SNMP OVERVIEW

Figure 6-2 provides a context diagram of the SNMP operator interface.

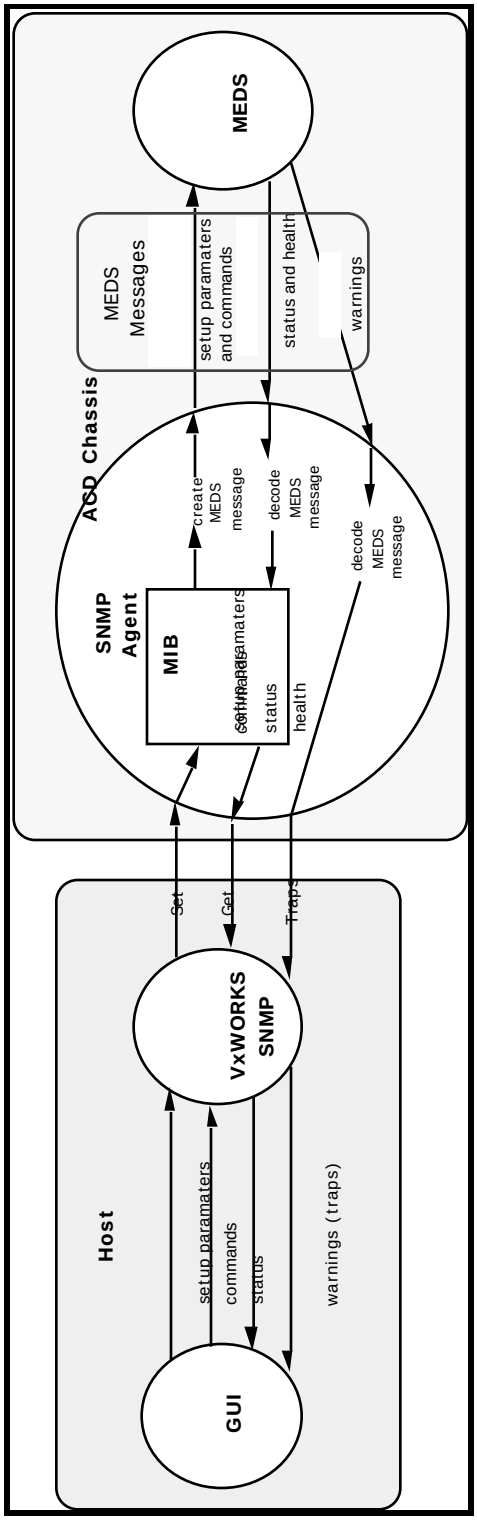


Figure 6-2. SNMP Interface

6.4.1 SNMP CONFIGURATION

To verify SNMP configuration, follow these steps:

a. cd "snmp/cfg" under ACD system root directory

b. check these files:

(1) agents - entries for each system plus vxatmcs1 (this is the default name used by the agents to look for configuration files)

(2) mgrs.v1 & mgrs.v2 - entries for each system that shall be running management software

c. If changes are made to these files, then v2config should be rerun

(1) execute: v2config noauth

(2) creates configs subdirectory containing agent and manager configuration files

(3) copy default snmpd.conf file into each agent subdirectory "ex. cp snmpd.conf configs/vxatmcs1/agt/"

(4) copy current snmpinfo.dat into each manager subdirectory "ex. cp ../../subsys/master/agent/src/snmpinfo.dat configs/vlsi13/mgr/"

(5) rename configs to vxacd

File definitions for snmpd.conf and snmpinfo.dat follow:

a. snmpd.conf: This file defines initial values for the system group, and whether authentication-failure traps should be generated.

b. snmpinfo.dat: This file contains information about all MIB variables which the Agent supports.

6.4.2 AGENT CONFIGURATION

In order for the agent to run on the ACD System, "/SNMPBASE" must exist and be valid. To verify that it exists, follow these steps:

a. Check the system boot script "acd.cmd" and make sure getdir has been called to create p521MountDir, p521MountMachine, & p521MountIp

b. Use the following command to create the mount point:

nfsMount p521MountMachine,p521MountDir,"/SNMPBASE"

c. Make sure that v2config has created the expected directories of the agent and manager in question

d. "/SNMPBASE/snmp/cfg/vxacd/vxatmcs1/agt" should contain the following (all but the last are auto-generated by v2config):

acl.pty

agt.pty

context.pty

view.pty

snmpd.cnf

6.4.3 SNMP TROUBLESHOOTING GUIDELINES

If the agent doesn't seem to work, here's some guidelines to help determine why

a. Check to see if the agent is running. The last part of the system boot script initializes the SNMP agent. If the agent is running after boot up, messages from the agent scroll across the screen as it identifies its current version and counts the subsystems. If these messages don't scroll across the screen, the agent never even comes up. To further verify that the agent is running, proceed as follows:

   (1) At the vxWorks prompt on the master, type" I"

   (2) A task list appears on the screen, and in the task list should be an entries for "_snmpd_main" and "_poll_MEDS_s"

   (3) If the entries are present, then the agent is running.

   (4) If the entries are not present and the messages never scrolled across the screen at bootup, then a VxWorks image that does not contain the agent has been loaded into the system software. Most likely, some just incorrectly upgraded the software.

b. Make sure network connection to system exists. To do this, ping the system from a workstation. If the ACD System cannot be pinged, then there's a network problem. Make certain that the network is attached.

c. See if SNMP communication can be established. Attempt a basic communication. Request the snmp system group from MIB-II using the following command line: "getmany vxedos2 public system" If a reply with the system variables is received, then the agent is at least running and communicating. If the request is unsuccessful, then there may be a problem with the Agent installation. Make sure the agent is running (see above).

d. Verify that the custom portion of agent is working. To do this, request the MEDS response group from the ACD MIB with the "getmany vxedos2 vxatmcs1 medsRspGrp" request. A reply with MEDS response group variables indicates that the custom portion of the SNMP agent is running.

If the ACD System agent passes all of the above tests, then everything should be fine with the SNMP Agent, but the subsystems may be having problems. Check the subsystem's health and status. If there is no reply or an error response to the health requests, then there is either an authentication problem (check config files), or an SNMP Agent without the custom extensions has been loaded into the system.